OpenBlocks FW4

リリース *1.0.0*

ぷらっとホーム株式会社

2020 年 06 月 19 日

目 次

第 I 部　商標について **21**

第 II 部　使用にあたって **23**

第 III 部　WEB UI セットアップガイド **25**

第 1 章　はじめに **26**

1.1 VX1 向けパッケージ内容 27
1.1.1 VX1 本体 1 台 27
1.1.2 デバッグ用 USB コンソールケーブル 1 本 27
1.1.3 ご使用にあたって 1 部 27
1.1.4 AC アダプタ 1 本 27
1.1.5 放熱・設置ブラケット 1 個 28
1.2 各部の名称 (VX1 本体) 28
1.2.1 モデム種類判別 30
1.3 VX2 向けパッケージ内容 31
1.3.1 VX2 本体 1 台 31
1.3.2 USB Type-A - Micro USB ケーブル 1 本 31
1.3.3 ご使用にあたって 1 部 31
1.3.4 AC アダプタ 1 本 31
1.3.5 放熱・設置ブラケット 1 個 32
1.4 各部の名称 (VX2 本体) 32
1.4.1 モデム種類判別 34
1.5 IX9 向けパッケージ内容 35
1.5.1 IX9 本体 1 台 35
1.5.2 USB Type-A - Micro USB ケーブル 1 本 35
1.5.3 ご使用にあたって 1 部 35
1.5.4 AC アダプタ 1 本 35
1.5.5 LAN ケーブル 1 本 36
1.6 各部の名称 (IX9 本体) 36
1.7 BX1 向けパッケージ内容 38
1.7.1 BX1 本体 1 台 38
1.7.2 USB 給電コンソールケーブル 1 本 38
1.7.3 ご使用にあたって 1 部 38
1.8 各部の名称 (BX1 本体) 39
1.9 BX3 向けパッケージ内容 40
1.9.1 BX3 本体 1 台 40
1.9.2 USB 給電コンソールケーブル 1 本 40

1.9.3 ご使用にあたって 1 部 40
1.10 各部の名称 (BX3 本体) 41
1.11 EX1 向けパッケージ内容 42
1.11.1 EX1 本体 1 台 42
1.11.2 USB Type-A - Micro USB ケーブル 1 本 42
1.11.3 ご使用にあたって 1 部 42
1.12 各部の名称 (EX1 本体) 43
1.12.1 モデム種類判別 46
1.13 BX0 向けパッケージ内容 47
1.13.1 BX0 本体 1 台 47
1.13.2 USB 給電二又ケーブル/Ethernet 付き 1 本 47
1.13.3 ご使用にあたって 1 部 47
1.14 各部の名称 (BX0 本体) 48
1.15 BX5 向けパッケージ内容 49
1.15.1 BX5 本体 1 台 49
1.15.2 USB 給電コンソールケーブル 1 本 49
1.15.3 ご使用にあたって 1 部 49
1.16 各部の名称 (BX5 本体) 50
1.17 ステータスインジケーター 51
1.18 放熱・設置ブラケット取付方法について 52

第 2 章 ご利用の前に 53
2.1 SIM について 53
2.2 OpenBlocks シリーズの設置 53
2.3 WEB クライアントの準備 54
2.3.1 WLAN 接続の場合 55
2.3.2 Ethernet 接続の場合 55

第 3 章 WEB UI の初期基本設定 57
3.1 使用許諾画面 57
3.2 管理者アカウント (WEB UI の管理者アカウント) 設定 58
3.3 ネットワーク設定画面 58
3.3.1 モバイルルーター構成 59
3.3.2 サーバ構成 63
3.3.3 WLAN AP モードの詳細設定 (CH 設定と国際対応) 66
3.3.4 Enterprise 認証について 68
3.4 内部時計設定 70
3.5 システム再起動による設定項目の反映 71
3.6 管理者ログイン画面 74
3.7 ダッシュボード画面 74

第 4 章 SMS コントロール 76
4.1 SMS コントロールの起動設定 76
4.2 SMS コントロールのコマンド 77
4.3 SMS での複数コマンド送信 78
4.4 SMS ユーザ定義スクリプトの登録 78
4.5 SMS コントロールコマンドのダイレクト実行 80

第 5 章　サービス機能 **81**
5.1 BT I/F 制御 82
5.2 状態 82
5.3 BT 登録 83
5.4 BLE 登録 84
5.5 EnOcean 登録 86
5.6 Modbus2(C) 登録 88
5.7 Modbus2(S) 登録 89
5.8 Modbus(C) 登録 90
5.9 Modbus(S) 登録 91
5.10 User デバイス登録 92

第 6 章　AirManage 機能 **93**
6.1 AirManage 初回アクセス設定 94
6.2 AirManage 登録 96
6.2.1 AirManage アカウント作成 96
6.2.2 テナント作成 100
6.2.3 本体の登録及びテナントへの所属 101

第 7 章　拡張機能 **103**
7.1 機能拡張用パッケージのインストール 103

第 8 章　設定項目別リファレンス **107**
8.1 サービス制御・拡張・AirManage 機能の表示/非表示 107
8.2 プロセス状況表示機能 108
8.3 ストレージアラート機能 109
8.4 root パスワードの設定 110
8.5 フィルター許可 110
8.6 SSH の鍵交換 113
8.7 WEB 管理者パスワード変更 116
8.8 WEB ユーザー 117
8.9 ファイル管理 117
8.10 ソフトウェアライセンスの表示 119
8.11 本体シリアルの確認 120
8.12 ダイナミック DNS 121
8.13 静的ルーティングの追加 122
8.14 通信確認 122
8.15 ネットワーク状態確認 123
8.16 コンフィグレーションのバックアップとリストア 125
8.17 システムソフトウェアのアップデート 126
8.18 SMS 送信 126
8.19 SSH トンネル 127
8.20 サポート情報 130
8.21 OpenBlocks のサービス及び技術情報一覧 132
8.22 FUNC スイッチの機能割当 133
8.23 監視機能 134
8.24 URI プロキシ機能 139

8.25 WEB コンソール機能 140
8.26 SYSLOG 転送機能 140
8.27 ストレージクリーンナップ機能 141
8.28 電源監視機能 142
8.29 本体自動再起動機能 143
8.30 Pub キー追加機能 144
8.31 HTTP プロキシ (クライアント) 機能 144
8.32 停止・再起動 146
8.33 フォーマット機能 146
8.34 簡易マウント機能 147
8.35 FSTAB 編集機能 148
8.36 ホスト追加機能 149
8.37 稼働情報機能 151

第 9 章 注意事項及び補足 153
9.1 OpenBlocks IoT VX シリーズの電源について 153
9.2 自動再起動機能 153
9.3 LTE/3G モジュール (ソフトバンク) 運用時のアクセス 153
9.4 GRUB メニュー表示方法について 153
9.5 Factory Reset(工場出荷状態への切り替え) 155
9.6 使用ポート一覧 155
9.7 自動外部ストレージマウント機能 156

第 IV 部 開発者向けガイド 157

第 10 章 はじめに 158

第 11 章 ご利用の前に 159
11.1 製品概要 159
11.2 SSD ベースのシステム開発における注意点 159
11.3 SIM について 161
11.4 eMMC ストレージパーティション情報 161
11.5 ストレージモード 162
11.6 PC と OpenBlocks の接続 163
11.7 WEB UI 164

第 12 章 ご利用方法 165
12.1 ステータスインジケーターの表示色 165
12.2 モバイル回線向けモデム制御 168
12.3 モバイル回線接続 172
12.3.1 オンデマンド接続 172
12.3.2 モバイル回線モデムのユーザー制御 174
12.4 バックアップ 176
12.5 リストア 178
12.6 応用 178
12.7 推奨使用デバイスファイル 179
12.8 Factory Reset(工場出荷状態への切り替え) 179

12.9 リカバリー起動 180
12.10 クロス開発環境の構築 183
12.11 WEB UI 拡張機能 183
12.11.1 WEB UI への機能拡張インストール 183
12.11.2 スクリプト編集 184
12.11.3 コマンド実行 185
12.11.4 SMS 送信 185
12.11.5 LTE モジュール (KDDI) 運用切り替え 186
12.11.6 WEB UI 自動外部ストレージマウント機能 186
12.11.7 ブートモード切替 187
12.11.8 GPS 情報について 187
12.11.9 DHCP における DGW、DNS について 188
12.11.10 Docker について 188

第 13 章 製品仕様 190
13.1 OpenBlocks IoT VX1 本体仕様 190
13.2 OpenBlocks IoT VX2 本体仕様 191
13.3 OpenBlocks IoT VX シリーズオプション 192
13.3.1 LTE/3G モジュール (ソフトバンク) 192
13.3.2 LTE モジュール (KDDI) 192
13.3.3 LTE モジュール (NTT ドコモ/KDDI) 193
13.3.4 BWA モジュール 193
13.3.5 EnOcean モジュール 193
13.3.6 Wi-SUN モジュール 194
13.3.7 LoRaWAN モジュール 194
13.4 OpenBlocks IoT VX2 オプション 194
13.4.1 内蔵バッテリー 194
13.4.2 ミスター省エネモジュール 195
13.5 OpenBlocks IX9 本体仕様 196
13.6 OpenBlocks IoT BX1 本体仕様 197
13.7 OpenBlocks IoT BX3 本体仕様 198
13.8 OpenBlocks IoT BX5 本体仕様 199
13.9 OpenBlocks IoT BX0 本体仕様 200
13.10 OpenBlocks IoT EX1 本体仕様 201
13.11 OpenBlocks IoT EX1 オプション 202
13.11.1 3G モジュール (NTT ドコモ) 202
13.11.2 3G モジュール (ソフトバンク) 202
13.11.3 LTE/3G モジュール (ソフトバンク) 202
13.11.4 LTE モジュール (KDDI) 203
13.11.5 LTE モジュール (NTT ドコモ/KDDI) 203
13.11.6 BWA モジュール 203
13.11.7 EnOcean モジュール 204
13.11.8 Wi-SUN モジュール 204
13.11.9 LoRaWAN モジュール 204
13.11.10 内蔵バッテリー 205
13.11.11 ミスター省エネモジュール 205

第 14 章 注意事項及び補足 **207**
14.1 スクリプト処理による遅延処理等について 207
14.2 使用ポート一覧 207
14.3 OpenBlocks IX9 の Mini DisplayPort について 208

第 V 部　データハンドリングガイド **209**

第 15 章 はじめに **210**

第 16 章 IoT データ制御機能 **211**

第 17 章 サービス機能の基本設定 **212**

第 18 章 IoT データ制御パッケージのインストール **213**

第 19 章 IoT データ制御のアプリ設定 **214**
19.1 アプリ起動制御 214
19.2 アプリ自動再起動設定 219

第 20 章 送受信設定（PD Repeater） **220**
20.1 送受信設定 220
20.1.1 送受信先毎の設定 222
本体内 (lo) 222
DEXPF(dexpf) 223
DEXPF[Websocket](dexpf_ws) 224
MS Azure IoT Hub(iothub) 225
MS Azure IoT Hub[Websocket](iothub_ws) 228
AWS IoT(awsiot) 231
AWS IoT[Websocket](awsiot_ws) 234
Google IoT Core(iotcore) 237
Watson IoT for Gateway(w4g) 240
MS Azure Event hubs(eventhub) 243
Amazon Kinesis(kinesis) 244
Watson IoT for Device(w4d) 245
Toami for DOCOMO(t4d) 248
SoftBank スマ可視専用クラウド (sbiot) 253
KDDI IoT クラウド Standard(kddi_std) 255
PH 社独自仕様 WEB サーバー (pd_web) 256
WEB サーバー (web) 258
MQTT サーバー (mqtt) 261
TCP(ltcp) 264
PD Exchange(pd_ex) 265
ドメインソケット (lsocket) 266
20.1.2 ビーコン送信設定 267
20.1.3 BLE デバイス情報送信設定 271
20.1.4 EnOcean デバイス設定 273
20.1.5 Wi-SUN B ルート情報送信設定 275
20.1.6 Modbus クライアントデバイス設定 277

20.1.7 Modbus サーバーデバイス設定 . . . 285
20.1.8 Modbus2 クライアントデバイス設定 . . . 289
20.1.9 Modbus2 サーバデバイス設定 . . . 297
20.1.10 SW4x デバイス設定 . . . 302
20.1.11 高圧スマートメータ (B ルート) 情報送信設定 . . . 305
20.1.12 デバイス設定 (ユーザー定義) . . . 312
20.1.13 下流方向制御 . . . 314
下流方向制御の概要 . . . 314
PD Repeater の下流方向メッセージ . . . 316
Modbus クライアント／サーバーの下流方向制御 . . . 319
Modbus クライアント . . . 321
Modbus サーバ . . . 324
Modbus 2 クライアント／サーバーの下流方向制御 . . . 327
Modbus 2 クライアント . . . 327
Modbus 2 サーバー . . . 329
SW4x の下流方向制御 . . . 330
PD Agent . . . 331
ユーザー定義デバイスの登録と送受信先の設定 . . . 331
PD Agent の設定 . . . 332
PD Agent の設定と制御メッセージ . . . 334
環境変数への継承 . . . 336
応答メッセージ . . . 337
デバイス制御方式に依存した送信メッセージ構文 . . . 338
MS Azure IoT Hub の送信メッセージ構文 . . . 338
AWS IoT の送信メッセージ構文 . . . 339
MQTT サーバーの送信メッセージ構文 . . . 339
20.1.14 IoT データ制御ツールログについて . . . 340
20.1.15 センサーデータの確認 . . . 342

第 21 章 カスタマイズ **344**
21.1 独自開発データ収集アプリケーション . . . 344
21.1.1 ユーザー定義デバイスの登録と送受信先の設定 . . . 344
21.1.2 PD Repeater へのデータ書き込み . . . 345
21.2 独自開発下流方向制御アプリケーション . . . 347
21.2.1 ユーザー定義デバイスの登録と送受信先の設定 . . . 347
21.2.2 PD Repeater からのデータ書き込み . . . 347
21.3 独自開発アプリケーションの起動／停止制御 . . . 348
21.3.1 アプリケーションの登録 . . . 348
21.3.2 アプリケーションが用いるスクリプトの指定 . . . 348
21.4 deb パッケージ . . . 349
21.5 アプリケーション設定の確認 . . . 349
21.6 複雑な構成の実現 . . . 350
21.7 PD Broker . . . 356
21.8 Lua 拡張 . . . 358
21.8.1 BLE Lua . . . 358
21.8.2 EnOcean Lua . . . 360
21.9 調整 . . . 362

21.10CSV データ送信機能 364

第 22 章 補足事項 **369**
22.1 データ送信量及び回線速度について 369
22.2 PD Repeater への書き込みデータフォーマット 369
22.3 PD Repeater へのデータの書き込みサイズ 370
22.4 PD Repeater のバッファーサイズ 370
22.5 Repeater のエラー時の再送信 370
22.6 独自開発アプリケーションの設定ファイルについて 370
22.7 Node-RED へのデータ経由方法について 370
22.8 BLE デバイスとして追加したビーコンについて 371
22.9 WEB サーバーへのデータ送信について 371
22.10Handler コンフィグユーザー設定 371
22.11PH 社独自仕様 Web サーバー（PD Web） 372
22.11.1 PD Web の概要 372
22.11.2 PD Web の HTTP ヘッダー 373
22.11.3 PD Web のトークン 374
22.11.4 Web サーバー (PHP スクリプト) の実装例 374
22.12IoT Hub / IoT Edge へのデータ送信について 379
22.13PD Exchange とアプリケーション、デバイスの紐づけ 380
22.14各送受信先クラウドにおけるプロトコルについて 381
22.15ダッシュボードからの起動・停止について 382

第 VI 部 Node-RED スターターガイド **383**

第 23 章 はじめに **384**

第 24 章 Node-RED 事前準備 **385**
24.1 Node-RED のインストール 385
24.2 Node-RED 起動設定 386
24.3 Node-RED のブラウザフィルタについて 387
24.4 パケットフィルタについて 388

第 25 章 Node-RED の簡易説明 **389**
25.1 Node-RED 画面構成 390
25.2 ノード種類 391
25.3 共通ノード 392
25.4 機能ノード 392
25.5 ネットワークノード 393
25.6 入力ノード 393
25.7 出力ノード 393
25.8 シーケンスノード 394
25.9 パーサノード 394
25.10ストレージノード 394
25.11ソーシャルノード 395
25.12cloud ノード 395
25.13GetwayKit ノード 395

25.14location ノード 396
25.15dashboard ノード 397
25.16Google ノード 397

第 26 章 ノード操作サンプル **398**
26.1 位置情報の可視化 398
26.2 ビーコンデータへの位置情報付与 403

第 27 章 その他 **410**
27.1 Node-RED ログについて 410
27.2 Node-RED へのノード追加 411
27.3 Node-RED の Config 編集 413
27.4 Node-RED のプロセス状況について 414
27.5 Node-RED の HTTPS 化 415
27.6 Node-RED のバージョンアップ 417

第 VII 部 Samba 設定ガイド **419**

第 28 章 はじめに **420**

第 29 章 Samba 設定 **421**
29.1 Samba のインストール 421
29.2 Samba 起動設定 422
29.3 Samba パケットフィルタについて 424

第 30 章 その他 **425**
30.1 外部ストレージとの連携について 425
30.2 Samba 起動設定 425

第 VIII 部 セキュリティガイド **426**

第 31 章 はじめに **427**

第 32 章 セキュリティ設定 **428**
32.1 セキュリティ機能のインストール 428
32.2 セキュリティ使用設定について 429
32.3 不正攻撃のアクセス拒否解除 430

第 33 章 その他 **432**
33.1 機能拡張等について 432

第 IX 部 カメラ設定ガイド **433**

第 34 章 はじめに **434**

第 35 章 カメラ機能 **435**
35.1 カメラ機能のインストール 435
35.2 カメラ機能設定 436

35.3 カメラ ストリーミング機能 436
35.4 動体検知機能 439
35.4.1 検知イベントトリガー機能 443
35.5 メディア機能 446
35.6 ログ機能 447

第 36 章 その他 **448**
36.1 カメラ及びストリーミングデータ確認用ユーザー 448
36.2 OpenBlocks IoT VX2 のカメラ接続について 448
36.3 VIDEO デバイスファイルについて 449
36.4 OpenBlocks IoT BX/EX シリーズへのカメラ設定について 449

第 X 部 Docker 設定ガイド **450**

第 37 章 はじめに **451**

第 38 章 Docker 設定 **452**
38.1 WEB UI 込み Docker のインストール 452
38.2 WEB UI からの Docker 使用について 453
38.3 Docker コンテナ一覧 454
38.4 Docker コンテナリソース表示 455
38.5 Docker コンテナリソース設定更新 456
38.6 Docker コンテナ初回起動 457
38.7 Docker ローカルイメージ確認 458
38.8 Docker イメージの検索 458
38.9 Docker イメージのダウンロード 459
38.10Docker コンテナのログ確認 460
38.11Docker コンテナのネットワーク設定 461
38.12プライベートレジストリサーバーの認証設定 463
38.13ボリューム一覧表示及び削除 464

第 39 章 その他 **465**
39.1 Azure IoT Edge との連携 465
39.2 OpenBlocks IX9 での使用について 465

第 XI 部 Azure IoT Edge 設定ガイド **466**

第 40 章 はじめに **467**

第 41 章 Azure IoT Edge 設定 **468**
41.1 WEB UI 込み Azure IoT Edge のインストール 468
41.2 [事前準備]IoT Hub の設定 469
41.3 WEB UI からの Azure IoT Edge 設定について 470
41.4 Azure IoT Edge のセットアップ 470
41.5 Azure IoT Edge 操作 471
41.6 Azure IoT Edge モジュールの環境変数 473

第 42 章 その他 **475**

42.1 WEB UI 用 Docker との連携 . . . 475
42.2 Azure IoT Edge のコンテナ追加 . . . 475
42.3 Azure IoT Edge のコンテナ追加 . . . 475
42.4 OpenBlocks IX9 での使用について . . . 475

第 XII 部　FTP 設定ガイド 476

第 43 章 はじめに 477

第 44 章 FTP 設定 478
44.1 FTP のインストール . . . 478
44.2 FTP サーバー起動設定 . . . 479
44.3 FTP パケットフィルタについて . . . 481
44.4 FTP ダウンロード設定 . . . 482
44.5 FTP アップロード設定 . . . 484

第 45 章 その他 488
45.1 外部ストレージとの連携について . . . 488
45.2 ストレージクリーナップ機能との連携について . . . 488
45.3 FTP クライアントにおけるファイル名について . . . 488

第 XIII 部　データハンドリング設定リファレンスガイド 490

第 46 章 共通事項 491
46.1 モジュール . . . 491
46.2 設定ファイル . . . 491
46.2.1 フォーマット . . . 491
46.2.2 デォルト値 . . . 491
46.3 モジュール間通信 . . . 491
46.3.1 モジュール間通信メソッド . . . 491
46.3.2 ソケットのパス名 . . . 491
46.3.3 データーサイズ . . . 492
46.3.4 モジュール間通信に関わるモジュール共通のキー . . . 492

第 47 章 PD Repeater 493
47.1 デォルトパス . . . 493
47.2 設定ファイルの書式 . . . 494
47.2.1 構文 . . . 494
47.2.2 ルートオブジェクト . . . 494
47.2.3 servers オブジェクト . . . 495
PD Excheng(pd_ex) . . . 496
汎用 MQTT ブローカー (mqtt) . . . 497
IBM Watson IoT for device (w4d) . . . 498
Amazon Kinesis (kinesis) . . . 499
PostgreSQL (pgsql) . . . 499
MySQL (mysql) . . . 500
汎用の Web サーバ (web) . . . 500

Amazon AWS IoT (awsiot) . . . 501
Microsoft Azure Event Hubs (eventhub) . . . 501
IBM Watson IoT for gateway (w4g) . . . 502
Microsoft Azure IoT Hub(iothub) . . . 503
NTT docomo Toami for docomo (t4d) . . . 504
UNIX ドメインソケット (lsocket) . . . 504
KDDI IoT クラウド Standard(kddi_std) . . . 505
NIFTY IoT Device Hub (nf_dvhub) . . . 505
Plat'Home 独自仕様 Web サーバ (pd web) . . . 506
Google Cloud IoT Core (iotcore) . . . 507
TCP ソケット (ltcp) . . . 507
Amazon AWS IoT [Websocket](awsiot_ws) . . . 508
Microsoft Azure IoT Hub[Websocket] (iothub_ws) . . . 509
ダミー出力 (dummy) . . . 509
SoftBank IoT (sbiot) . . . 510
Things クラウド (things) . . . 511
FTP サーバ (ftp) . . . 512
47.2.4 device オブジェクト . . . 513
PD Exchenge(pd_ex) . . . 515
汎用の MQTT ブローカー (mqtt) . . . 515
IBM Watson IoT for device (w4d) . . . 515
Amazon Kinesis (kinesis) . . . 515
PostgreSQL (pgsql) . . . 516
MySQL (mysql) . . . 516
汎用の Web サーバ (web) . . . 516
Amazon AWS IoT (awsiot) . . . 516
Microsoft Azure Event Hubs (eventhub) . . . 517
IBM Watson IoT for gateway (w4g) . . . 517
Microsoft Azure IoT Hub(iothub) . . . 517
NTT docomo Toami for docomo (t4d) . . . 517
UNIX ドメインソケット (lsocket) . . . 518
KDDI IoT クラウド Standard (kddi std) . . . 518
NIFTY IoT Device Hub (nf_dvhub) . . . 518
Plat'Home 独自仕様 Web サーバ (pd web) . . . 518
Google Cloud IoT Core (iotcore) . . . 519
TCP ソケット (ltcp) . . . 519
Amazon AWS IoT[Websocket](awsiot_ws) . . . 519
Microsoft Azure IoT Hub[Websocket](iothub_ws) . . . 519
ダミー出力 (dummy) . . . 520
SoftBank IoT(sbiot) . . . 520
Things クラウド (things) . . . 520
FTP サーバ (ftp) . . . 520
47.3 下流方向メッセージ . . . 522
47.4 Plat ’Home 独自仕様 WEB サーバ . . . 524
47.4.1 HTTP ヘッダー . . . 524
47.4.2 トークン . . . 524

第 48 章 PD Broker **525**
48.1 デォルトパス 525
48.2 設定ファイルの書式 525
48.2.1 構文 525
48.2.2 broker オブジェクト 526
48.2.3 destinations オブジェクト 526

第 49 章 PD Agent **527**
49.1 デォルトパス 527
49.2 設定ファイルの書式 527
49.2.1 構文 527
49.2.2 agent オブジェクト 528
49.2.3 channels オブジェクト 529
49.2.4 実行オブジェクトに継承される環境変数 530
49.2.5 Dynamic Link モジュール 531
49.2.6 Lua 言語スクリプト 535
49.2.7 応答メッセージ 538

第 50 章 PD Handler BLE (Node.js) **539**
50.1 デォルトパス 539
50.2 設定ファイルの書式 539
50.2.1 構文 539
50.2.2 ルートオブジェクト 539
50.2.3 servers オブジェクト 540
50.2.4 beacon オブジェクト 540
50.2.5 payload_manage オブジェクト 540
50.2.6 data_filter_rule オブジェクト 541
50.2.7 blesensor オブジェクト 541

第 51 章 PD Handler BLE (C) **542**
51.1 デォルトパス 542

第 52 章 PD Handler UART **543**
52.1 デォルトパス 543
52.2 設定ファイルの書式 543
52.2.1 EnOcean 543
構文 543
ルートオブジェクト 543
enocean オブジェクト 544
enocean_device オ ブジェクト 544
52.2.2 Wi-SUN B ルート 544
構文 544
ルートオブジェクト 544
wisun オブジェクト 545

第 53 章 PD Handler HVSMC **546**
53.1 デォルトパス 546
53.2 設定ファイルの書式 546

53.2.1 構文 546
53.2.2 ルートオブジェクト 546
53.2.3 hvsmc オブジェクト 547

第 54 章 PD Handler Modbus **548**
54.1 Modbus ファンクションコード 548
54.2 PD Handler Modbus Client 549
54.2.1 デォルトパス 549
54.2.2 設定ファイルの書式 550
構文 550
ルートオブジェクト 550
clients オブジェクト 551
acquisitions オブジェクト 552
54.2.3 利用可能なファンクションコード 553
54.2.4 CSV ファイル 554
54.2.5 基準時刻制御 555
54.2.6 メッセージオブジェクト 556
54.2.7 クラウドからの制御 557
54.3 PD Handler Modbus Server 558
54.3.1 デォルトパス 558
54.3.2 設定ファイルの書式 559
構文 559
ルートオブジェクト 559
servers オブジェクト 560
54.3.3 利用可能なファンクションコード 561
54.3.4 クラウドからからの制御 562
54.3.5 レジスタマップのダンプと作成 563

第 55 章 PD Handler Modbus 2 **565**
55.1 Modbus モデリングファイル 565
55.1.1 インデックスの階層 566
55.1.2 モデリングファイルの例 566
55.1.3 Modbus モデリングファイルの書式 569
構文 569
ルートオブジェクト 570
maker オブジェクト 571
product オブジェクト 571
model オブジェクト 571
sku オブジェクト 572
bit オブジェクト 572
register オブジェクト 573
rawRequest オブジェクト 573
bitsArray オブジェクト 574
registersArray オブジェクト 574
55.1.4 モデリングファイル作成支援ツール 574
構文確認ツール 575
マージツール 576

55.2 PD Handler Modbus 2 Client . . . 578
55.2.1 デォルトパス . . . 578
55.2.2 設定ファイルの書式 . . . 579
構文 . . . 579
ルートオブジェクト . . . 579
clients オブジェクト . . . 580
acquisitions オブジェクト . . . 581
55.2.3 CSV ファイル . . . 582
55.2.4 基準時刻制御モード . . . 583
55.2.5 メッセージオブジェクト . . . 584
55.2.6 クラウドからの制御 . . . 585
55.2.7 CLI (pd_handler_modbus_2_cli) . . . 587
55.3 PD Handler Modbus 2 Server . . . 589
55.3.1 デォルトパス . . . 589
55.3.2 設定ファイルの書式 . . . 590
構文 . . . 590
ルートオブジェクト . . . 590
servers オブジェクト . . . 591
55.3.3 メッセージオブジェクト . . . 593
55.3.4 クラウドからの制御 . . . 594

第 56 章 PD Handler SW4x **595**
56.1 対応ノード一覧 . . . 595
56.2 デォルトパス . . . 596
56.3 モデリングファイルの書式 . . . 597
56.3.1 構文 . . . 597
56.3.2 モデルキー . . . 597
56.3.3 モデルオブジェクト . . . 598
56.3.4 keys オブジェクト . . . 599
56.3.5 nodeInfos オブジェクト . . . 599
56.3.6 inits オブジェクト . . . 600
56.3.7 polls オブジェクト . . . 601
汎用 . . . 601
SW-4280-1000 Modbus RTU ノード用 . . . 601
56.3.8 モデル別オブジェクト . . . 602
SW4X_COMMON オブジェクト . . . 602
SW42P0_x01 オブジェクト . . . 604
SW4210_1202 オブジェクト . . . 604
SW4210_1205 オブジェクト . . . 605
SW4210_1204 オブジェクト . . . 605
SW4220_1000 オブジェクト . . . 606
SW4240_1000 オブジェクト . . . 606
SW4220_1020 オブジェクト . . . 607
SW42J0_1202 オブジェクト . . . 608
SW42K0_1000 オブジェクト . . . 609
SW42DD_1000 オブジェクト . . . 610
SW42DD_1100 オブジェクト . . . 612

SW42D0_1000 オブジェクト 614
SW4230_1000 オブジェクト 615
SW4260_1x00 オブジェクト 616
SW4280_1000 オブジェクト 617
SW4260_1x20 オブジェクト 618
SW4000_1000, SW4100_1000, SB4020_1000, SW4300_1000, SW4500_1000 オブジェクト 620
56.4 RTU モデリングファイルの書式 621
56.5 Modbus モデリングファイルの書式 621
56.6 基準時刻制御モード 622
56.7 設定ファイルの書式 623
56.7.1 構文 623
56.7.2 設定オブジェクト 624
56.8 メッセージオブジェクト 625
56.8.1 コマンド入力オブジェクト 625
56.8.2 応答メッセージオブジェクト 626
56.9 クラウドからの制御 627
56.10CLI(pd_handler_sw4x_cli) 627

第 XIV 部　PD Handler JSON フォーマット一覧 628

第 57 章 PD Handler BLE (Node.js) 629
57.1 Beacon 629
57.1.1 ビーコン送受信設定 629
57.1.2 BLE デバイス情報送信設定 630
57.2 Sensor 631
57.2.1 TI Sensor 631
BLE デバイス情報送信設定 631
57.2.2 Fujitsu Sensor 633
ビーコン送受信設定 633
BLE デバイス情報送信設定 634
57.2.3 ALPS IoT Smart Module 635
ビーコン送受信設定 635
BLE デバイス情報送信設定 637
57.2.4 オムロン 環境センサ 641
ビーコン送受信設定 641
BLE デバイス情報送信設定 643
57.2.5 オムロン USB 型環境センサ 646
ビーコン送受信設定 646
BLE デバイス情報送信設定 647
57.2.6 ユニ電子 BLE 温湿度センサー (Logtta) 648
ビーコン送受信設定 648
BLE デバイス情報送信設定 649
57.2.7 ユニ電子 BLE　CO2 センサー (Logtta CO2) 650
ビーコン送受信設定 650
BLE デバイス情報送信設定 651

57.2.8 ユニ電子 BLE　水温センサー (Logtta Water) 652
ビーコン送受信設定 652
BLE デバイス情報送信設定 653
57.2.9 ラトックシステム Bluetooth ホコリセンサー 654
BLE デバイス情報送信設定 654
57.2.10 ラトックシステム Bluetooth エアクオリティモニター 655
BLE デバイス情報送信設定 655
57.2.11 ラトックシステム ワットチェッカー 656
BLE デバイス情報送信設定 656
57.2.12 エレックス工業 μ PRISM 657
BLE デバイス情報送信設定 657
57.2.13 EnOcean マルチセンサ (EMSIB) 659
ビーコン送受信設定 659
BLE デバイス情報送信設定 660
57.2.14 EnOcean 照度センサ&人感センサ (EMDCB) 661
ビーコン送受信設定 661
BLE デバイス情報送信設定 662

第 58 章 PD Handler BLE (C) with Lua 663
58.1 Beacon 663
58.2 Sensor 663
58.2.1 ナカヨ 呼出しボタン 664
ビーコン送受信設定 664
BLE デバイス情報送信設定 665

第 59 章 PD Handler UART 666
59.1 EnOcean with Lua 666
59.1.1 人感センサー (EEP : A50701) 666
59.1.2 あけしめセンサー (EEP : D50001) 667
59.1.3 温度センサー (EEP : A50205) 667
59.1.4 温湿度センサー (EEP : A50402) 668
59.1.5 温湿度センサー (EEP : A50403) 668
59.1.6 2 相式 CT センサー (EEP : A51201) 669
59.1.7 3 相式 CT センサー (EEP : D23202) 669
59.1.8 大気圧センサー (EEP : A50501) 670
59.1.9 照度センサー (EEP : A50601) 671
59.1.10 照度センサー (EEP : A50602) 672
59.1.11 照度センサー (EEP : A50605) 673
59.1.12 CO2 センサー (EEP : A50904) 674
59.1.13 デジタル入力センサー (EEP : A53005) 675
59.1.14 2 ロッカースイッチ (EEP : F60204) 676
59.1.15 マルチセンサ (EEP : D21441) 677
59.1.16 マルチセンサ (EEP : D21440) 678
59.1.17 温湿度センサー (EEP : A50401) 679
59.1.18 照度センサー (EEP : A50603) 680
59.1.19 マルチセンサ (EEP : A51405) 681
59.1.20 RAW データ時 682

59.2 3.2. Wi-SUN . . . 683
59.2.1 瞬時電力 (B ルート) . . . 683
59.2.2 積算電力 (B ルート) . . . 683

第 60 章 PD Handler HVSMC 685
60.1 RAW データ . . . 685
60.2 動作状態 . . . 686
60.3 異常発生状態 . . . 686
60.4 ECHONET Lite 属性情報 . . . 687
60.5 高圧スマート電力量メータ属性情報 . . . 688
60.6 定時計測値 . . . 689
60.7 計測値 . . . 690
60.8 需要電力 . . . 690

第 61 章 PD Handler Modbus 691
61.1 Modbus クライアント (Modbus マスター) . . . 692
61.1.1 PLC 機器へのポーリング動作時 . . . 692
TCP プロトコルによるレジスタ出力又はレジスタ入力の読み込み . . . 692
RTU プロトコルによるデジタル出力又はデジタル入力の読み込み . . . 693
61.1.2 クラウドからのオンデマンド動作時 . . . 694
TCP プロトコルによるレジスタ出力又はレジスタ入力の読み込み . . . 694
RTU プロトコルによるデジタル出力又はデジタル入力の読み込み . . . 696
TCP プロトコルによるレジスタ入力への書き込み . . . 698
RTU プロトコルによるデジタル入力への書き込み . . . 700
スレーブ ID の読み出し . . . 702
エラー時 . . . 704
61.2 Modbus サーバー (Modbus スレーブ) . . . 705
61.2.1 PLC 機器からの書き込み動作時 . . . 705
TCP プロトコルによるレジスタ入力への書き込み . . . 705
RTU プロトコルによるデジタル入力への書き込み . . . 706
61.2.2 クラウドからのオンデマンド動作時 . . . 707
レジスタ出力又はレジスタ入力の読み込み . . . 707
デジタル出力又はデジタル入力の読み込み . . . 709
レジスタ出力又はレジスタ入力への書き込み . . . 711
デジタル出力又はデジタル入力への書き込み . . . 713
スレーブ ID の読み出し . . . 715
エラー時 . . . 716

第 62 章 PD Handler Modbus 2 717
62.1 Modbus クライアント (Modbus マスター) . . . 718
62.1.1 PLC 機器へのポーリング動作時 . . . 718
62.1.2 クラウドからのオンデマンド動作時 . . . 719
ポーリング中のレジスタに対するオンデマンド読み込み . . . 719
レジスタに対するオンデマンド書き込み . . . 721
エラー発生時の応答メッセージ . . . 723
62.2 Modbus サーバー (Modbus スレーブ) . . . 724
62.2.1 PLC 機器からの書き込み動作時 . . . 724

レジスタ出力への書き込み 724
62.2.2 クラウドからのオンデマンド動作時 725
レジスタの読み込み 725
レジスタへの書き込み 727
エラー発生時の応答メッセージ 729

第 63 章 PD Handler SW4x **730**
63.1 SW42P0-1x01 温度センサーノード 731
63.2 SW4210-1202 温・湿度センサーノード 732
63.3 SW4210-1205 照度センサーノード 733
63.4 SW4210-1204 温・湿度・照度センサーノード 734
63.5 SW4220-1000 人感センサーノード（活動量） 735
63.6 SW4240-1000 パルスカウントノード 736
63.7 SW4220-1020 人感センサーノード（イベントドリブン仕様） 737
63.8 SW42J0-1202 リモコン温湿度ノード 738
63.9 SW42K0-1000 パルスピックノード 739
63.10 SW42DD-1000 電流センサーノード (5A) 740
63.11 SW42DD-1100 電流センサーノード (200A) 742
63.12 SW42D0-1000 電流センサーノード 744
63.13 SW4230-1000 CO2 ノード 746
63.14 SW4260-1x10 電力モニターノード 747
63.15 SW4280-1000 Modbus RTU ノード 748
63.16 SW4260-1x20 電力モニターノード 749
63.17 SW4100-1000 ルーターノード 750
63.18 ベースノード 751

第I部

商標について

- Linux は、Linus Torvalds 氏の米国およびその他の国における商標あるいは登録商標です。
- Firefox は、Mozilla Foundation の米国およびその他の国における登録商標です。
- Google Chrome は、Google Inc. の登録商標です。
- Internet Explorer は、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。
- Microsoft、.NET、Windows、Microsoft Azure、Internet Explorer は、米国 Microsoft Corporation の米国およびその他の国における商標または登録商標です。
- NTT ドコモは日本電信電話株式会社の登録商標です。
- SoftBank およびソフトバンクの名称、ロゴは、日本国およびその他の国におけるソフトバンクグループ株式会社の登録商標または商標です。
- au(KDDI) は KDDI 株式会社の登録商標または商標です。
- 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。
- Docker and Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.

第II部

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。
- 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。

第III部

WEB UIセットアップガイド

第1章　はじめに

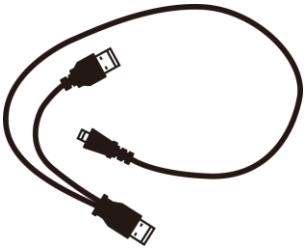
本書は、OpenBlocks シリーズを WEB ユーザーインターフェース (以下、WEB UI) で設定する方法を解説しています。本設定には、WEB ブラウザが使用可能なクライアント装置 (PC やスマートフォン、タブレット等) が必要になります。

1.1 VX1 向けパッケージ内容

OpenBlocks IoT VX1 の標準品構成は以下となります。

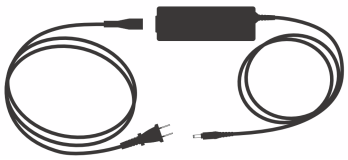
1.1.1 VX1 本体 1 台

1.1.2 デバッグ用 USB コンソールケーブル 1 本

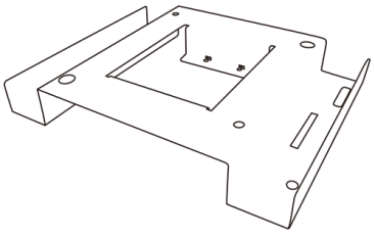


1.1.3 ご使用にあたって 1 部

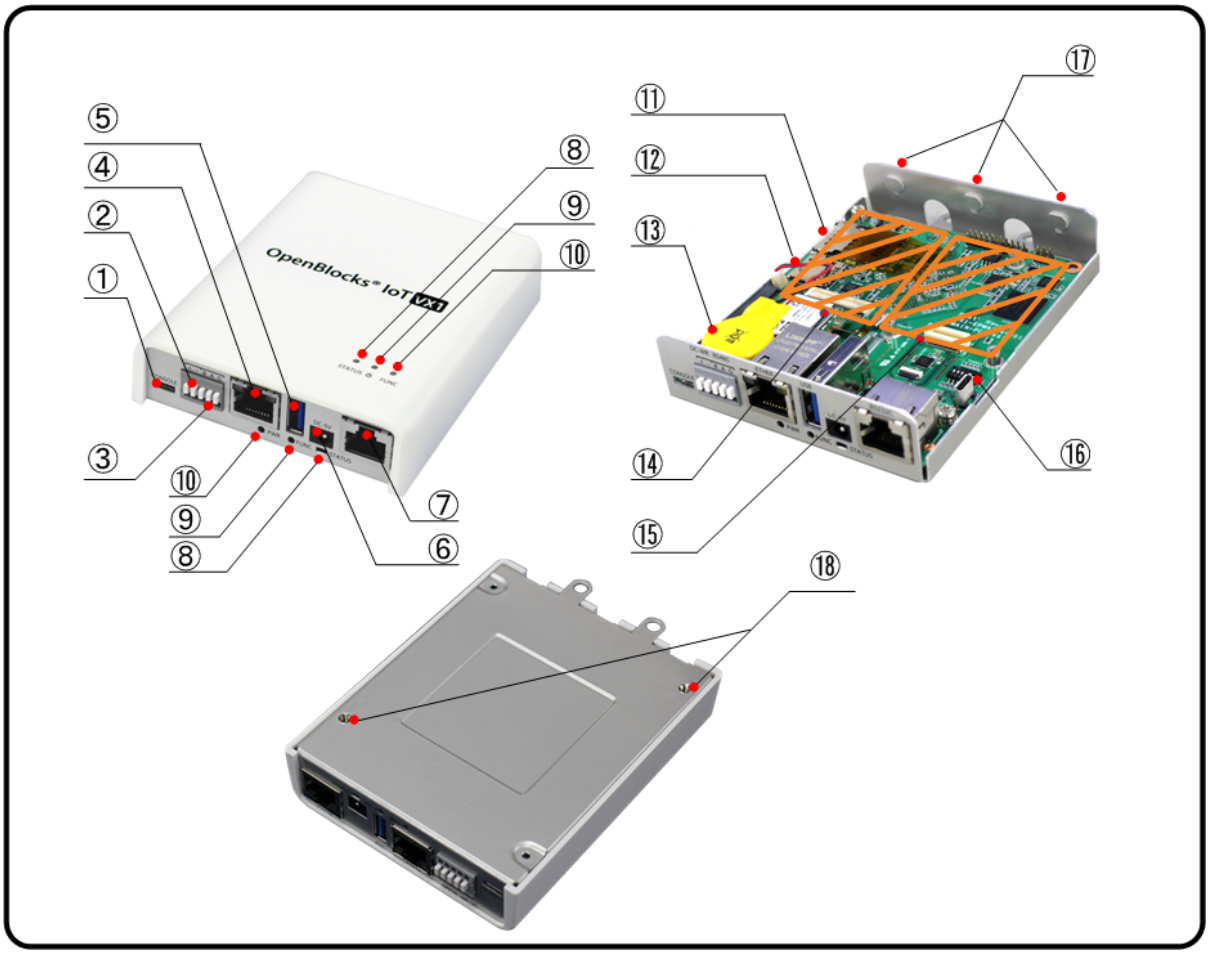
1.1.4 AC アダプタ 1 本



1.1.5 放熱・設置ブラケット 1 個



1.2 各部の名称 (VX1 本体)



No.	名称	備考
1	USB シリアルコンソールポート	Micro USB
1	ワイドレンジ電源入力	
3	RS-485(半二重) コネクタ	
4	イーサネットポート	10BASE-T / 100BASE-TX / 1000BASE-T
5	USB ホストモードポート	A-Type/USB3.0
6	専用 AC アダプタ入力	DC5V
7	RS-232C ポート	RJ-45。 オプションで D-Sub9 ピンとの接続コネクタを販売しています。 接続ケーブルは一般のストレートネットワークケーブルが利用できます。
8	ステータスインジケーター	7 色の LED で点灯、点滅をします。
9	パワースイッチ	OS 稼働中の場合、OS をシャットダウンします。未稼働状態の場合、OS が起動します。
10	FUNC スイッチ	割当された機能が動作します。
11	SIM スロット	SIM を挿入するスロットです。
12	MMC スロット	MMC はシステム運用に十分な信頼性を確保できない為、ファイル交換やログ保存用等にご利用ください。
13	RTC 用電池	
14	拡張スロット 1 用コネクタ	モバイル回線用のモバイルアダプタカードの拡張スロット用のコネクタです。使用するキャリア対応のモバイルアダプタカードを取り付けます。原則的に工場出荷オプションとなります。
15	拡張スロット 2 用コネクタ	EnOcean や Wi-SUN モジュール等の拡張スロット用のコネクタです。
16	DIP スイッチ	工場出荷オプションで設定されるので通常は変更しないでください。SW1〜3 : モデム判別用 SW4〜5 : 未使用 SW6 : OFF=RS485 ターミネータ ON(デフォルト)
17	外部アンテナ取付穴	画像では穴埋めされています。
18	放熱・設置ブラケット取付穴	

SIM の挿入は VX1 本体を裏返しにして SIM スロットの奥まで挿入してください。また、抜くときも同様に VX1 本体を裏返しにして取り出してください。

1.2.1 モデム種類判別

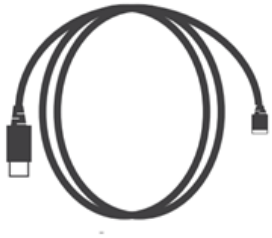
モデム種類	SW1	SW2	SW3
LTE/3G モジュール (ソフトバンク)	ON	OFF	OFF
LTE モジュール (KDDI)	ON	ON	OFF
LTE モジュール (NTT ドコモ)	ON	OFF	ON
LTE モジュール (NTT ドコモ/KDDI)	OFF	ON	ON
BWA モジュール	OFF	ON	ON
モデム未搭載	ON	ON	ON

1.3 VX2 向けパッケージ内容

OpenBlocks IoT VX2 の標準品構成は以下となります。

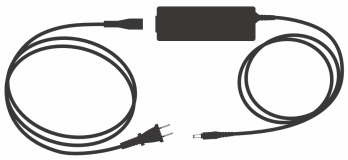
1.3.1 VX2 本体 1 台

1.3.2 USB Type-A - Micro USB ケーブル 1 本

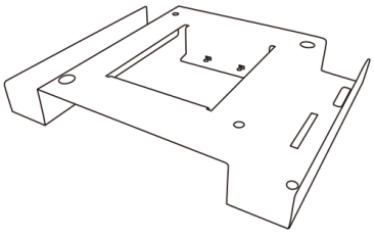


1.3.3 ご使用にあたって 1 部

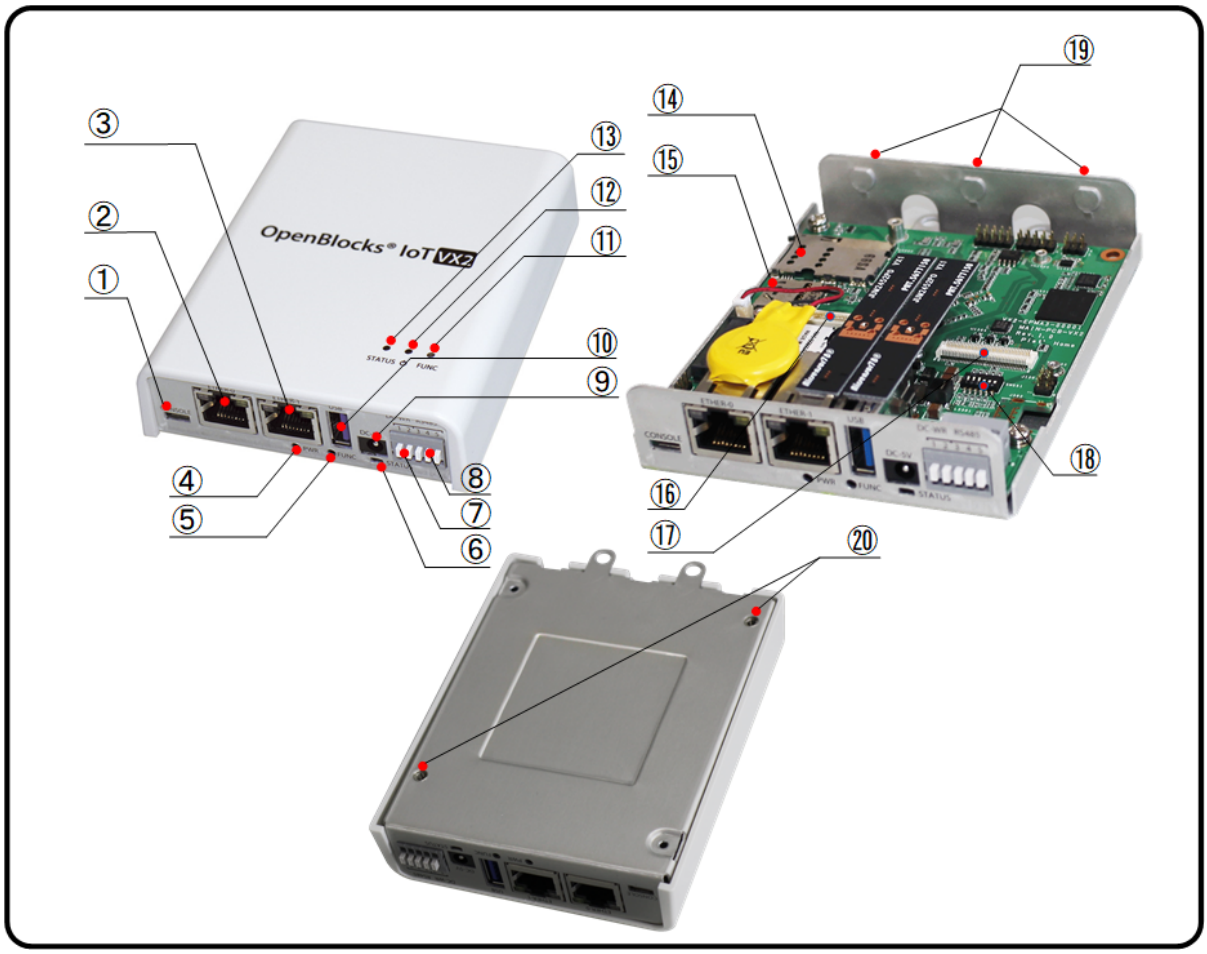
1.3.4 AC アダプタ 1 本



1.3.5 放熱・設置ブラケット 1 個



1.4 各部の名称 (VX2 本体)



No.	名称	備考
1	USB シリアルコンソールポート	Micro USB
2	イーサネットポート 0	10BASE-T / 100BASE-TX / 1000BASE-T
3	イーサネットポート 1	
4	パワースイッチ	OS 稼働中の場合、OS をシャットダウンします。未稼働状態の場合、OS が起動します。
5	FUNC スイッチ	割当された機能が動作します。
6	ステータスインジケーター	7 色の LED で点灯、点滅をします。
7	ワイドレンジ電源入力	"+"または"1" : DC プラス"-"または"2" : DC マイナス
8	RS-485(半 二重) コネクタ	"B"または"3" : 485B 端子"A"または"4" : 485A 端子"G"または"5" : GND
9	USB ホストモードポート	A-Type/USB3.0
10	専用 AC アダプタ入力	DC5V
11	FUNC スイッチ	割当された機能が動作します。
12	パワースイッチ	OS 稼働中の場合、OS をシャットダウンします。未稼働状態の場合、OS が起動します。
13	ステータスインジケーター	7 色の LED で点灯、点滅をします。
14	SIM スロット	SIM を挿入するスロットです。使用可能な SIM 形状は型番によって異なります。型番：OBSVX2/N 使用可能な SIM 形状：nano-SIM(4FF) 上記以外の場合使用可能な SIM 形状：mini-SIM(2FF)
15	MMC スロット	MMC はシステム運用に十分な信頼性を確保できない為、ファイル交換やログ保存用等にご利用ください。
16	拡張スロット 2 用コネクタ	EnOcean や Wi-SUN モジュール等の拡張スロット用のコネクタです。
17	拡張スロット 1 用コネクタ	モバイル回線用のモバイルアダプタカードの拡張スロット用のコネクタです。使用するキャリア対応のモバイルアダプタカードを取り付けます。原則的に工場出荷オプションとなります。
18	DIP スイッチ	工場出荷オプションで設定されるので通常は変更しないでください。SW1〜3 : モデム判別用 SW4〜5 : 未使用 SW6 : OFF=RS485 ターミネータ ON(デフォルト)
19	外部アンテナ取付穴	画像では穴埋めされています。
20	放熱・設置ブラケット取付穴	

SIM の挿入は VX2 本体を裏返しにして SIM スロットの奥まで挿入してください。また、抜くときも同様に VX2 本体を裏返しにして取り出してください。

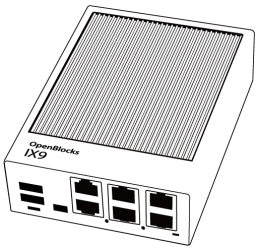
1.4.1 モデム種類判別

モデム種類	SW1	SW2	SW3
LTE/3G モジュール (ソフトバンク)	ON	OFF	OFF
LTE モジュール (KDDI)	ON	ON	OFF
LTE モジュール (NTT ドコモ)	ON	OFF	ON
LTE モジュール (NTT ドコモ/KDDI)	OFF	ON	ON
BWA モジュール	OFF	ON	ON
モデム未搭載	ON	ON	ON

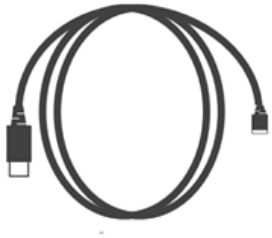
1.5 IX9 向けパッケージ内容

OpenBlocks IX9 の標準品構成は以下となります。

1.5.1 IX9 本体 1 台

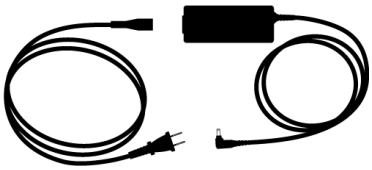


1.5.2 USB Type-A - Micro USB ケーブル 1 本



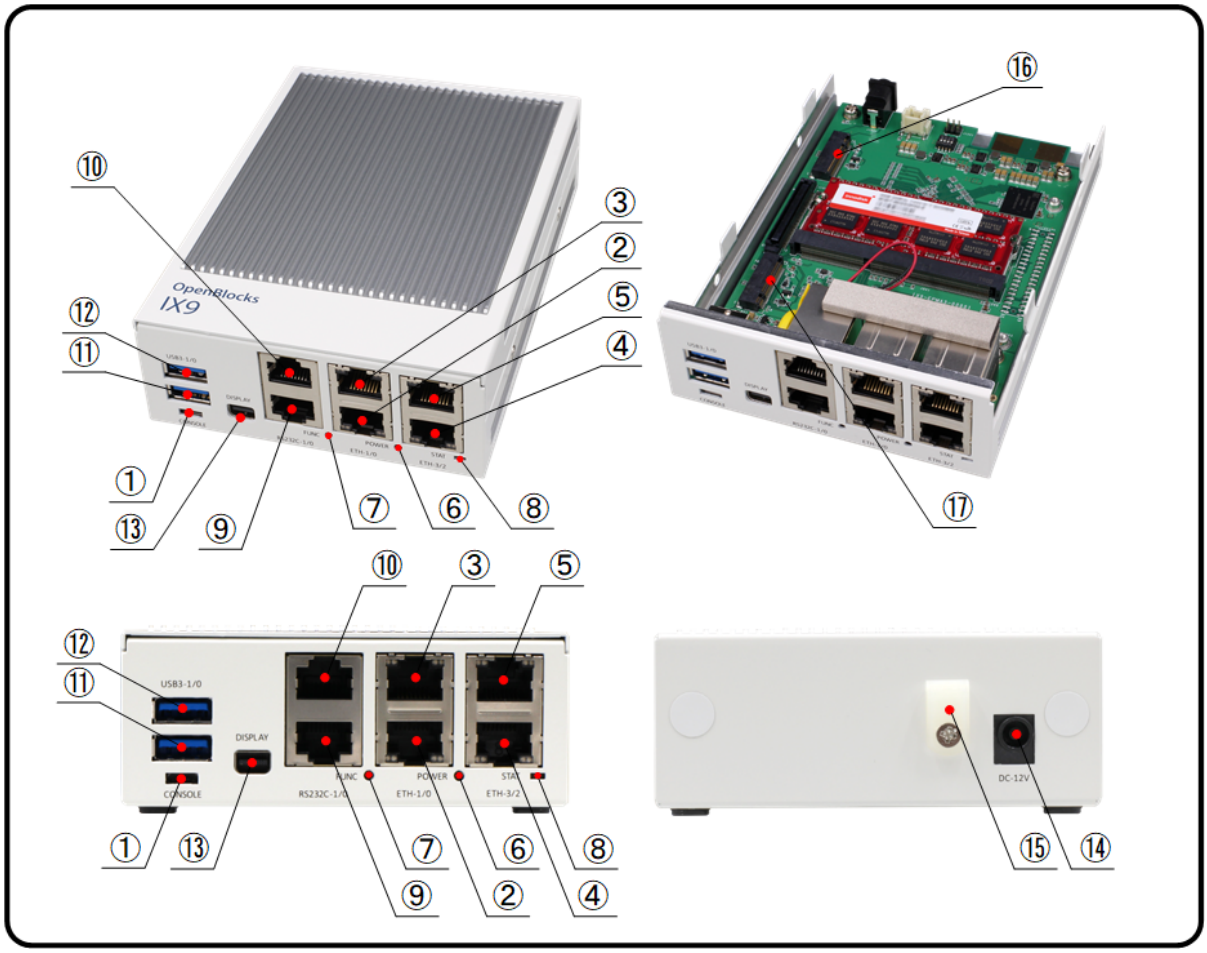
1.5.3 ご使用にあたって 1 部

1.5.4 AC アダプタ 1 本



1.5.5 LAN ケーブル 1 本

1.6 各部の名称 (IX9 本体)



No.	名称	備考
1	USB シリアルコンソールポート	Micro USB
2	イーサネットポート 0	10BASE-T / 100BASE-TX / 1000BASE-T
3	イーサネットポート 1	
4	イーサネットポート 2	
5	イーサネットポート 3	
6	パワースイッチ	OS 稼働中の場合、OS をシャットダウンします。未稼働状態の場合、OS が起動します。
7	FUNC スイッチ	割当された機能が動作します。
8	ステータスインジケーター	7 色の LED で点灯、点滅をします。
9	RS-232C ポート 0	RJ-45 オプションで D-Sub9 ピンとの接続コネクタを販売しています。接続ケーブルは一般のストレートネットワークケーブルが利用できます。
10	RS-232C ポート 1	
11	USB ホストモードポート 0	A-Type/USB3.0
12	USB ホストモードポート 1	
13	Mini DisplayPort	シングルモード。
14	専用 AC アダプタ入力	DV12V
15	クランプ	
16	M.2 Type 2280, key M 用スロット	SATA 3.0Gbps
17	M.2 Type 2280, key B 用スロット	PCIe Gen2, USB3.0, SIM, I2C

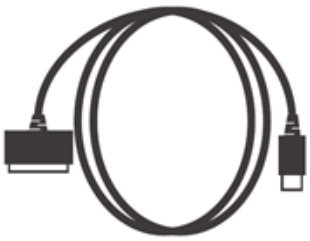
※ Mini DisplayPort はシングルモードの為、DVI 出力や HDMI 出力を行う場合にはアクティブタイプの変換アダプタが必用となります。尚、動作確認を行った変換アダプタは Plugable 社製「MDP-HDMI」となります。

1.7 BX1 向けパッケージ内容

OpenBlocks IoT BX1 の標準品構成は以下となります。

1.7.1 BX1 本体 1 台

1.7.2 USB 給電コンソールケーブル 1 本



1.7.3 ご使用にあたって 1 部

ver 1.0

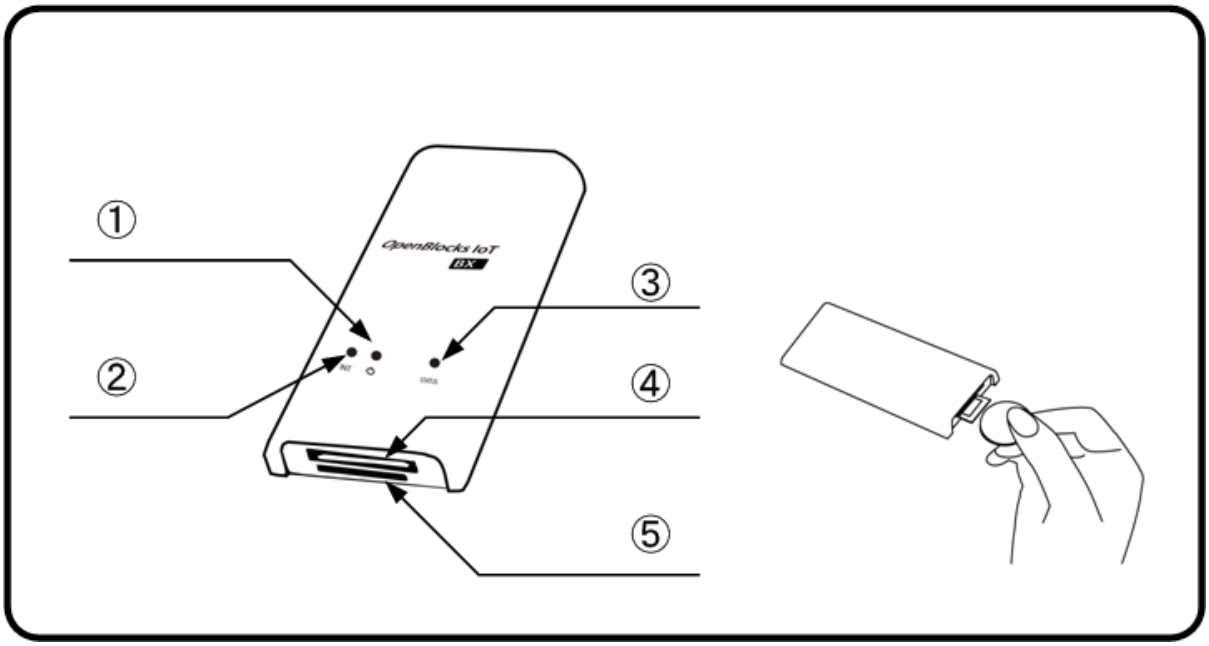
OpenBlocks IoT Family

ご使用にあたって

- 保証書
- ご使用上の注意
- 各種規定
- ソフトウェアについて

ご使用に当たって（本冊子・各種証書）は、製品をご使用される前に必ずご覧ください。
製品保証書、製品情報規定等詳細は以下URLをご参照ください。
http://www.plathome.co.jp/support/

1.8 各部の名称 (BX1 本体)



No.	名称	備考
1	パワースイッチ	短押しで OS をシャットダウンします。 (INIT スイッチの 4 秒以上長押しと同一) また、8 秒以上の長押しで強制的に電源 OFF します。
2	FUNC スイッチ	割当された機能が動作します。 尚、INIT スイッチと印字されている場合は、FUNC スイッチとして読み替えてください。
3	ステータスインジケーター	7 色の LED で点灯、点滅をします。
4	BX1 コネクタ	様々な IO に対応したコネクタです。 それぞれの IO に合わせたケーブルを接続可能です。
5	SIM スロット	3G 回線 (NTT ドコモ系列) を契約した SIM を挿入するスロットです。 ※対応する SIM の形状は mini-SIM(2FF) となります。(一般的に標準 SIM と呼ばれる規格)

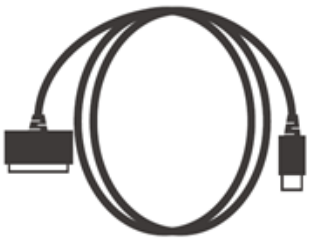
※ SIM の挿入はコネクタ面を上にし、コイン等を使って奥まで入れてください。また、抜くとき時も同様にコインを使用し、SIM スロットの奥まで差し込むとロックが外れてせり出してきます。

1.9 BX3 向けパッケージ内容

OpenBlocks IoT BX3 の標準品構成は以下となります。

1.9.1 BX3 本体 1 台

1.9.2 USB 給電コンソールケーブル 1 本



1.9.3 ご使用にあたって 1 部

ver 1.0

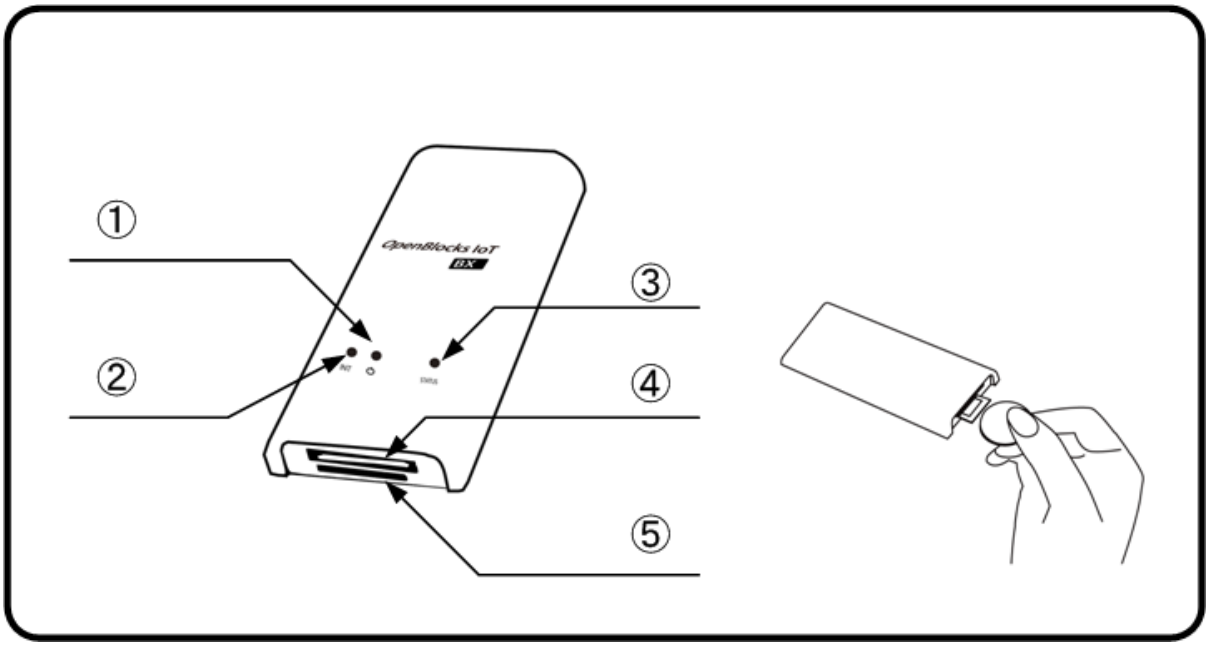
OpenBlocks IoT Family

ご使用にあたって

- 保証書
- ご使用上の注意
- 各種規定
- ソフトウェアについて

ご使用に当たって（本冊子・各種証書）は、製品をご使用される前に必ずご覧ください。
製品保証書、製品情報規定等は以下URL をご参照ください。
http://www.plathome.co.jp/support/

1.10 各部の名称 (BX3 本体)



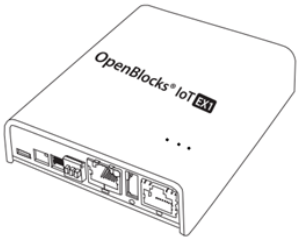
No.	名称	備考
1	パワースイッチ	短押しで OS をシャットダウンします。 (INIT スイッチの 4 秒以上長押しと同一) また、8 秒以上の長押しで強制的に電源 OFF します。
2	FUNC スイッチ	割当された機能が動作します。 尚、INIT スイッチと印字されている場合は、FUNC スイッチとして読み替えてください。
3	ステータスインジケーター	7 色の LED で点灯、点滅をします。
4	BX1 コネクタ	様々な IO に対応したコネクタです。 それぞれの IO に合わせたケーブルを接続可能です。
5	SIM スロット	3G 回線 (ソフトバンク系列) を契約した SIM を挿入するスロットです。 ※対応する SIM の形状は mini-SIM(2FF) となります。(一般的に標準 SIM と呼ばれる規格)

※ SIM の挿入はコネクタ面を上にし、コイン等を使って奥まで入れてください。また、抜くとき時も同様にコインを使用し、SIM スロットの奥まで差し込むとロックが外れてせり出してきます。

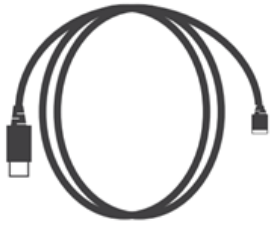
1.11 EX1 向けパッケージ内容

OpenBlocks IoT EX1 の標準品構成は以下となります。

1.11.1 EX1 本体 1 台



1.11.2 USB Type-A - Micro USB ケーブル 1 本



1.11.3 ご使用にあたって 1 部

ver 1.0

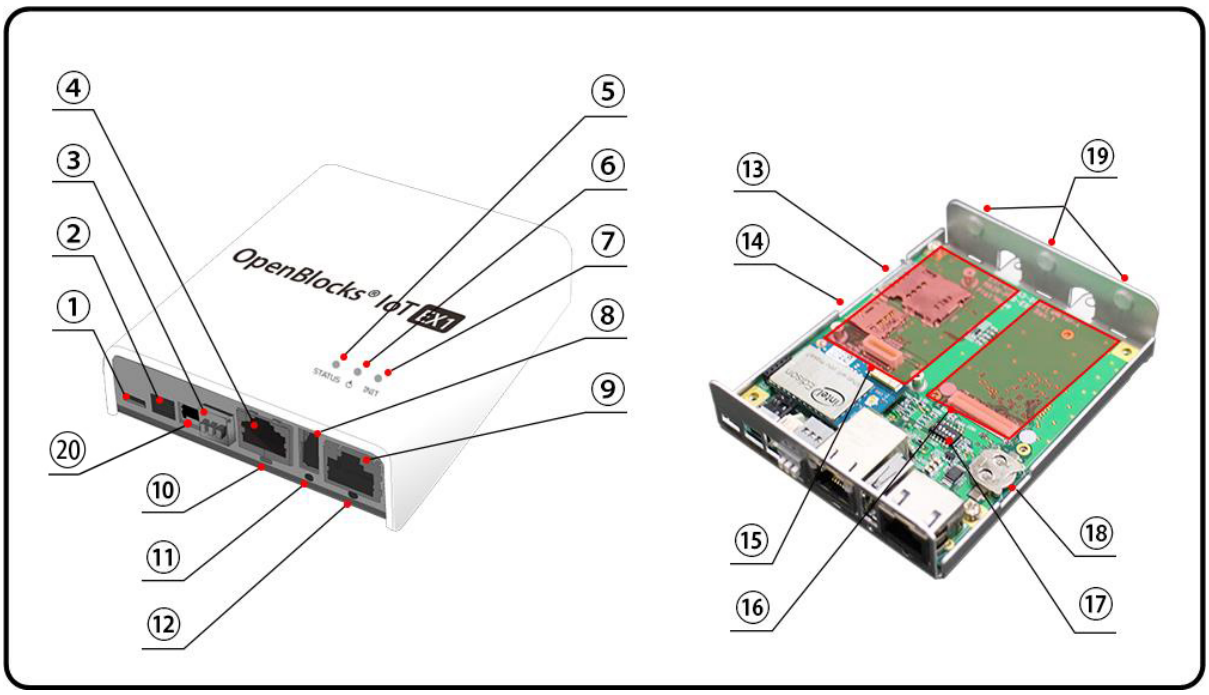
OpenBlocks IoT Family

ご使用にあたって

- 保証書
- ご使用上の注意
- 各種規定
- ソフトウェアについて

ご使用に当たって（本冊子・各種証書）は、製品をご使用される前に必ずご覧ください。
製品保証書、製品情報規定等詳細は以下URLをご参照ください。
http://www.plathome.co.jp/support/

1.12 各部の名称 (EX1 本体)



No.	名称	備考
1	USB シリアルコンソールポート	Micro USB。 バスパワーに対応した USB シリアルコンソールポートです。
2	専用 AC アダプタ入力	DC5V
3	RS-485(半二重)コネクタ	
4	イーサネットポート	100Base
5	ステータスインジケーター 1	7 色の LED で点灯、点滅をします。
6	パワースイッチ 1	短押しで OS をシャットダウンします。 (INIT スイッチ 1 及び 2 の 4 秒以上長押しと同一) また、8 秒以上の長押しで強制的に電源 OFF します。
7	FUNC スイッチ 1	割当された機能が動作します。 尚、INIT スイッチと印字されている場合は、FUNC スイッチとして読み替えてください。
8	USB ホストモードポート	A-Type
9	RS-232C ポート	RJ-45。 オプションで D-Sub9 ピンとの接続コネクタを販売しています。 接続ケーブルは一般のストレートネットワークケーブルが利用できます。
10	ステータスインジケーター 2	ステータスインジケーター 1 と同機能
11	パワースイッチ 2	パワースイッチ 1 と同機能
12	FUNC スイッチ 2	FUNC スイッチ 1 と同機能

No.	名称	備考
13	SIM スロット	SIM を挿入するスロットです。 使用可能な SIM 形状は型番によって異なります。 ●型番：OBSEX1G/N 使用可能な SIM 形状：nano-SIM(4FF) ●上記以外の場合 使用可能な SIM 形状：mini-SIM(2FF)
14	MMC スロット	MMC はシステム運用に十分な信頼性を確保できない為、ファイル交換やログ保存用等にご利用ください。
15	拡張スロット 1	EnOcean や Wi-SUN モジュール等の拡張スロットです。
16	拡張スロット 2	モバイル回線用のモバイルアダプタカードの拡張スロットです。 使用するキャリア対応のモバイルアダプタカードを取り付けます。原則的に工場出荷オプションとなります。
17	DIP スイッチ	工場出荷オプションで設定されるので通常は変更しないでください。 SW1 : 常時 ON SW4 : 未使用 SW6 : OFF=RS485 ターミネータ ON(デフォルト) ●モデル：OBSEX1 SW2/SW3 : モデム種類判別用 SW5 : OFF=RS-232C 使用 (デフォルト)、 ON=RS-485 使用 ●モデル：OBSEX1G SW2/SW3/SW5 : モデム種類判別用
18	RTC 用電池ホルダー	
19	外部アンテナ取付穴	画像では穴埋めされています。
20	ワイドレンジ電源入力	製品型番：OBSEX1 では使用不可。 DC 5〜48V 対応

SIM の挿入は EX1 本体を裏返しにして SIM スロットの奥まで挿入してください。また、抜くときも同様に EX1 本体を裏返しにして取り出してください。

1.12.1 モデム種類判別

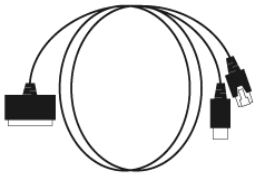
モデル	モデム種類	SW2	SW3	SW5
OBSEX1	3G モジュール	OFF	OFF	•
	LTE/3G モジュール (ソフトバンク)	OFF	OFF	•
	LTE モジュール (KDDI)	ON	OFF	•
	LTE モジュール (NTT ドコモ)	OFF	ON	•
	モデム未搭載	ON	ON	•
OBSEX1G	3G モジュール	OFF	OFF	OFF
	LTE/3G モジュール (ソフトバンク)	OFF	OFF	OFF
	LTE モジュール (KDDI)	ON	OFF	OFF
	LTE モジュール (NTT ドコモ)	OFF	ON	OFF
	LTE モジュール (NTT ドコモ/KDDI)	ON	OFF	ON
	BWA モジュール	ON	OFF	ON
	モデム未搭載	ON	ON	OFF

1.13 BX0 向けパッケージ内容

OpenBlocks IoT BX0 の標準品構成は以下となります。

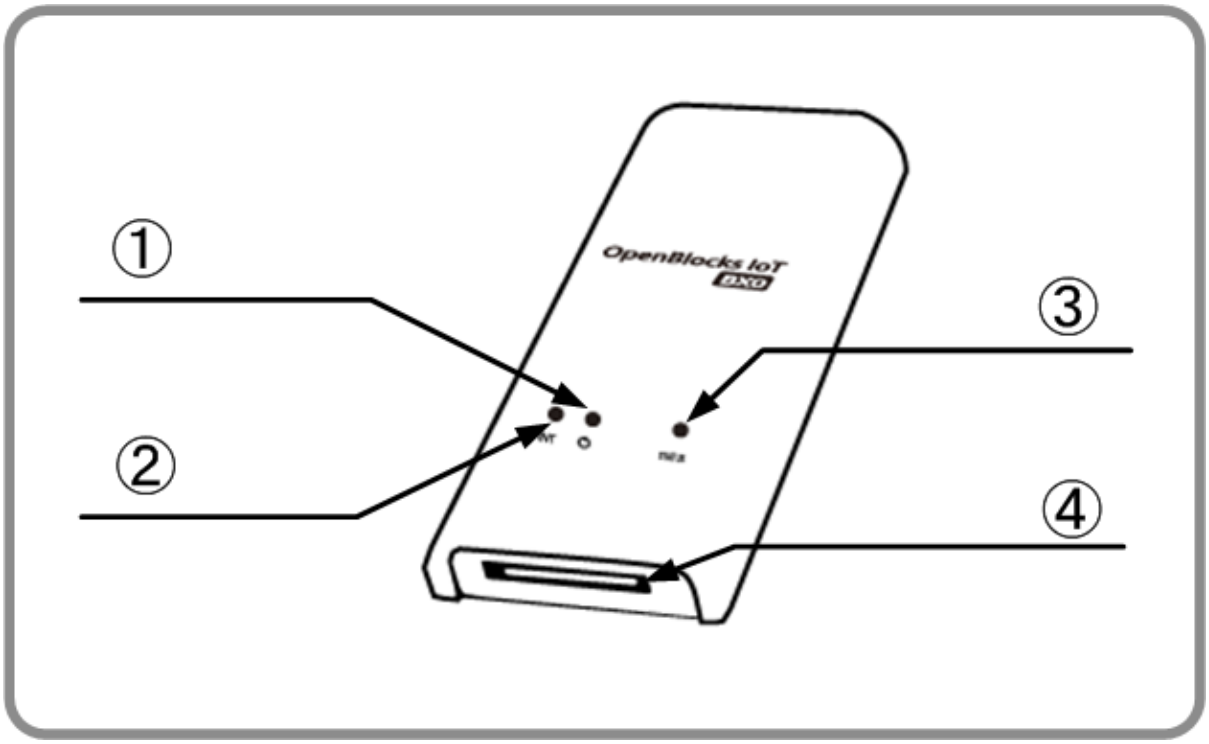
1.13.1 BX0 本体 1 台

1.13.2 USB 給電二又ケーブル/Ethernet 付き 1 本



1.13.3 ご使用にあたって 1 部

1.14 各部の名称 (BX0 本体)



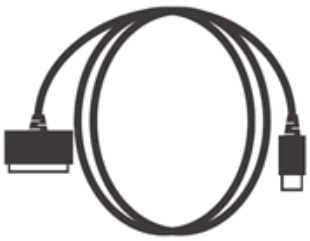
No.	名称	備考
1	パワースイッチ	短押しで OS をシャットダウンします。 (INIT スイッチの 4 秒以上長押しと同一) また、8 秒以上の長押しで強制的に電源 OFF します。
2	FUNC スイッチ	割当された機能が動作します。 尚、INIT スイッチと印字されている場合は、FUNC スイッチとして読み替えてください。
3	ステータスインジケーター	7 色の LED で点灯、点滅をします。
4	BX1 コネクタ	様々な IO に対応したコネクタです。 それぞれの IO に合わせたケーブルを接続可能です。

1.15 BX5 向けパッケージ内容

OpenBlocks IoT BX5 の標準品構成は以下となります。

1.15.1 BX5 本体 1 台

1.15.2 USB 給電コンソールケーブル 1 本



1.15.3 ご使用にあたって 1 部

ver 1.0

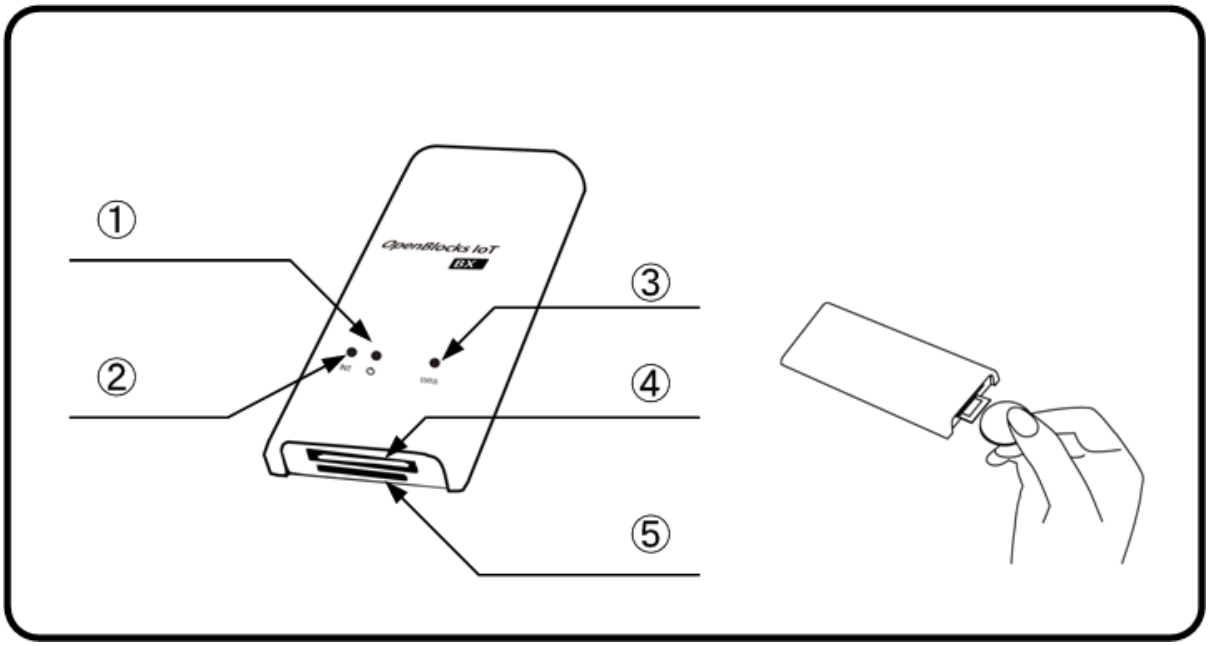
OpenBlocks IoT Family

ご使用にあたって

- 保証書
- ご使用上の注意
- 各種規定
- ソフトウェアについて

ご使用に当たって（本冊子・各種証書）は、製品をご使用される前に必ずご覧ください。
製品保証書、製品情報規定等詳細は以下URLをご参照ください。
http://www.plathome.co.jp/support/

1.16 各部の名称 (BX5 本体)



No.	名称	備考
1	パワースイッチ	短押しで OS をシャットダウンします。 (INIT スイッチの 4 秒以上長押しと同一) また、8 秒以上の長押しで強制的に電源 OFF します。
2	FUNC スイッチ	割当された機能が動作します。 尚、INIT スイッチと印字されている場合は、FUNC スイッチとして読み替えてください。
3	ステータスインジケーター	7 色の LED で点灯、点滅をします。
4	BX1 コネクタ	様々な IO に対応したコネクタです。 それぞれの IO に合わせたケーブルを接続可能です。
5	SIM スロット	LTE 回線 (NTT ドコモ/KDDI または地域 BWA 系列) を契約した SIM を挿入するスロットです。 ※対応する SIM の形状は mini-SIM(2FF) となります。(一般的に標準 SIM と呼ばれる規格)

※ SIM の挿入はコネクタ面を上にし、コイン等を使って奥まで入れてください。また、抜くとき時も同様にコインを使用し、SIM スロットの奥まで差し込むとロックが外れてせり出してきます。

1.17 ステータスインジケーター

本装置のステータスインジケーターは 7 色の LED で状態を表示します。以下が、各状態を表す状態となります。

状態	色	点灯状態	備考
本体及び OS 起動中	黄	点灯	本体起動及び OS 起動が終わるとモバイル回線の電波受信チェックへ移行します。※ SIM が挿入されていない場合は緑点滅。
SIM スロット未使用	緑	点滅	SIM が無い状態での正常稼働または電波受信待機状態への移行待ち状態。
モバイル回線電波：強	白	点滅	「開発者ガイド」の電波状態詳細参照。
モバイル回線電波：中	水色	点滅	「開発者ガイド」の電波状態詳細参照。
モバイル回線電波：弱	青	点滅	「開発者ガイド」の電波状態詳細参照。この電波強度での通信はリトライが多発する可能性があります。そのため、モバイル回線を使用する場合にはなるべく電波強度が中以上の状態にて使用してください。
モバイル回線電波：圏外	紫	点滅	「開発者ガイド」の電波状態詳細参照。
FUNC ボタンによる機能有効時	黄	点滅	モバイル回線や SIM スロット未使用時におけるステータスインジケーターと交互点滅となります。
OS 終了中	黄	点灯	
AirManage 初回アクセス失敗時	赤	点灯	AirManage リモート管理サーバへの初回アクセスが失敗した際に表示となります。WEB UI 未使用時の場合は 5 分後に OS が終了します。
強制 SIM モード時におけるモデムデバイスファイルまたは SIM カード認識失敗時	赤	点灯	モデム搭載モデルにおいて強制 SIM モードを有効にし、起動時にモデムのデバイスファイルが存在しないまたは SIM カードが認識できない場合の表示となります。5 分後に OS が再起動します。
AirManage 及び SIM 認識失敗時の OS 終了中	赤	点灯	

1.18 放熱・設置ブラケット取付方法について

OpenBlocks IoT VX2 及び EX1 への放熱・設置ブラケットの取付方法については、「放熱・設置ブラケット取扱説明書」をご参照ください。

第2章　ご利用の前に

2.1 SIM について

OpenBlocks シリーズにて、搭載可能な SIM 形状は一部の型番の物を除き mini-SIM(2FF) です。mini-SIM 対応モデルにて micro-SIM 及び nano-SIM を使用する場合には、脱落防止フィルム有及び接着テープ有で SIM を固定できるアダプタを使用してください。尚、SIM アダプタを使用した場合での SIM スロットの破損は有償修理対象となります為、ご注意ください。

2.2 OpenBlocks シリーズの設置

OpenBlocks IoT Family(OpenBlocks IoT VX シリーズは除く) は USB 充電器を外部バスパワー電源として利用するので別途お買い求めください。（USB 充電器は PSE マーク付きの国内安全規格品をご利用ください。また、出力電力は 1A 以上の物を使用してください。）添付の USB 給電コンソールケーブルを使い本装置と USB 充電器を接続します。

OpenBlocks IoT VX2 及び IX9 では付属の AC アダプタを用いて以下のように接続します。

※ OpenBlocks IoT VX2 ではワイドレンジ電源入力についてもサポートしておりますので、そちらでの稼働についてもご検討下さい。

利用可能状態になるとステータスインジケーターが点灯・点滅します。

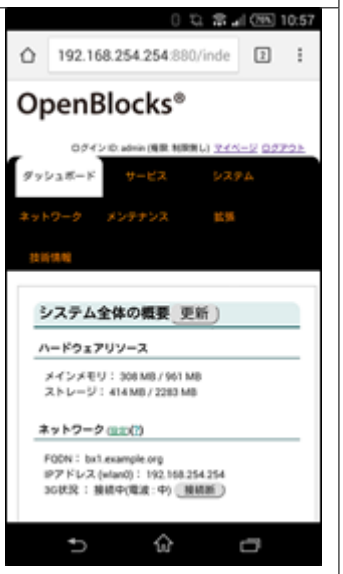
（表示色はその時の状態によります。）

2.3 WEB クライアントの準備

注釈: WEB クライアントは日本語設定にて、WEB UI へアクセスしてください。

本装置の WEB UI にアクセスするには、WEB クライアントが必要です。WEB クライアントには Ethernet 使用可能または WLAN 接続可能な PC やタブレット、スマートフォンが利用できます。WLAN 設定時には本装置のアクセスポイント (SSID) を選択し接続します。

2.3.1 WLAN 接続の場合

SSID 接続時	WEB 画面
	

スナップショットはスマートフォンの画面で、WLAN の SSID 一覧から本装置の SSID("iotfamily_"本体シリアル番号) を選択した画面です。ここで出荷時デフォルトのパスワード"openblocks"と入力すると接続できます。WLAN 接続できたら WEB ブラウザを使い次のアドレスにアクセスします。※本体シリアル番号は筐体の背面に記載されています。

	WLAN 接続時 URL
HTTP 接続	http://192.168.254.254:880
HTTPS 接続	https://192.168.254.254:4430

2.3.2 Ethernet 接続の場合

接続する WEB クライアント (PC) の Ethernet ポートと OpenBlocks シリーズの eth0 ポートを LAN ケーブルにて直接接続します。

WEB クライアントの IP アドレスを 169.254.0.0 のネットワークにアクセスできる IP アドレス (169.254.0.100 等の 200 以外) を設定し WEB ブラウザにて次のアドレスにアクセスしてください。

※ WEB クライアントが Windows マシンでかつ、対象の Ethernet インターフェースが DHCP クライアント設定となっている場合には 169.254.0.0 のアドレスが自動で設定されます。そのため、IP アドレスの設定は不要です。

	Etherrnet 接続時 URL
HTTP 接続	http://169.254.0.200:880
HTTPS 接続	https://169.254.0.200:4430

※パソコンでの WEB クライアントとして用いる WEB ブラウザは Google Chrome 及び Firefox の最新バージョンをサポートします。また、Internet Explorer では一切の操作が行えませんのでご使用しないでください。

第3章　WEB UIの初期基本設定

スマートフォン上の WEB ブラウザでも本設定は可能ですが、本書ではパソコンの WEB 画面を用いて解説を行います。

使用許諾画面 項から **ネットワーク設定画面** 項は工場出荷状態の時に必要な手順なので、それ以外の時は **内部時計設定** 項からの手順を参照ください。また、**ネットワーク設定画面** 項までが本装置を初期設定するために必要な最小限の手順で、モバイルルーター的な設定、または単体サーバとしての最小限のネットワーク設定が説明されています。

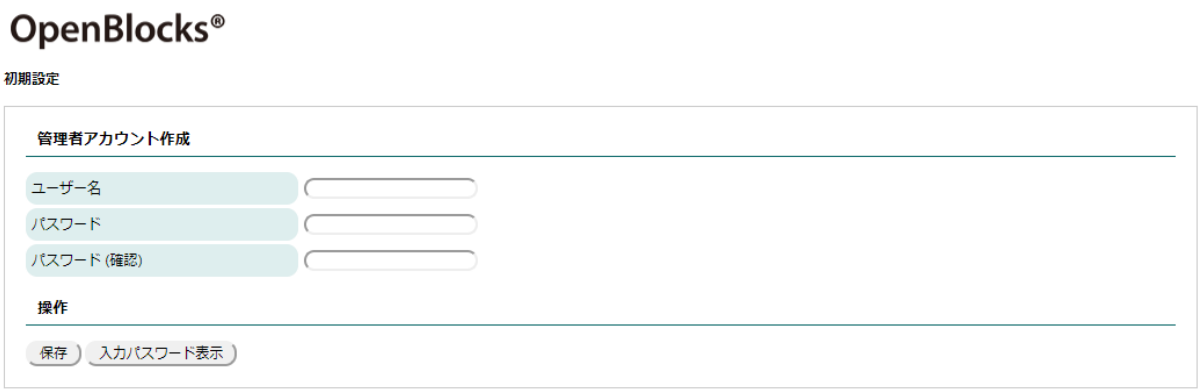
注釈: 本章にて実施している **管理者アカウント** *(WEB UI* **の管理者アカウント***)* **設定** 項での管理者アカウントの設定はセキュリティ上重要です。その為、クラックされにくくなるようなパスワードを設定してください。

3.1 使用許諾画面

本装置に何も設定されていない出荷直後では、本装置における使用許諾契約書の画面が表示されます。この使用許諾に合意出来る場合のみ本装置を利用することが出来ます。

画面をスクロールして契約内容を確認の上で、「同意する」を選択して次の画面に進みます。「同意しない」を選択した場合には、Google へリダイレクトされます。

3.2 管理者アカウント (WEB UI の管理者アカウント) 設定



使用許諾契約書に同意いただいた場合、WEB UI の管理者アカウントとパスワード入力画面が開きます。入力中のパスワードを表示させるには「入力パスワード表示」を押してください。

注釈: ここで入力する管理者のユーザ名は後で変更できない為間違わないように入力してください。このアカウントは root ユーザのパスワード変更権限を持つ為、注意してください。

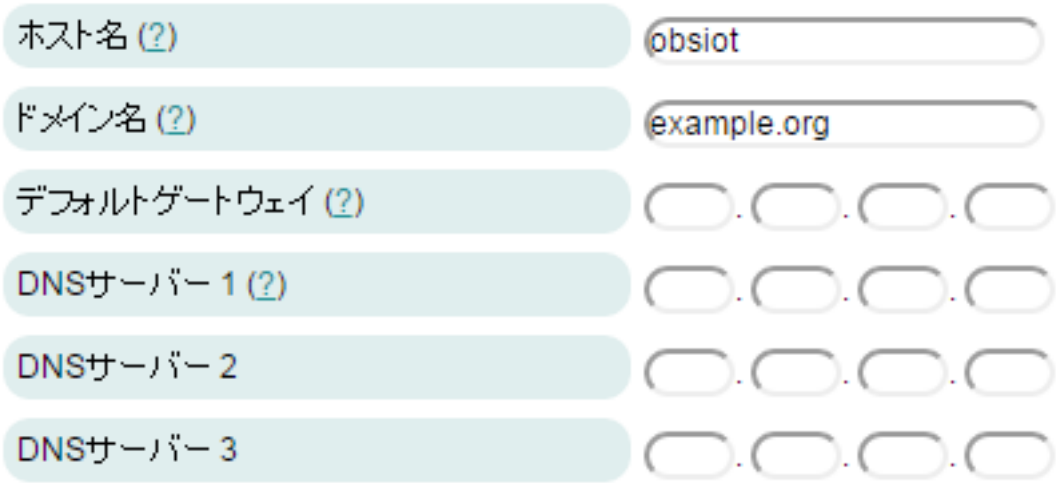
アカウント情報を設定し、保存ボタンを押すと最初のコンフィグレーション情報が書き込まれます。コンフィグレーションが書き込まれますと、次回のアクセスからは **使用許諾画面** 項と **管理者アカウント** *(WEB UI* **の管理者アカウント***)* **設定** 項の画面は表示されなくなり、WEB アクセスでの最初の画面は管理者のログイン画面が表示されます。

3.3 ネットワーク設定画面

OpenBlocks シリーズを利用する時に最小限の設定が必要なネットワーク設定画面です。

OpenBlocks IoT VX2 にてモデムモジュールを搭載している製品を用いて説明を行います。本装置をモバイルルーターとして使う構成、本装置をサーバ装置としてモバイル回線を使わない構成の二通りあります。

下図の通り、ネットワーク設定の基本タブの上の部分に本装置の名前を入力する欄があります。



項目	説明
ホスト名	本装置のサーバとしての名前です。
ドメイン名	本装置の所属するネットワークドメイン名です。
デフォルトゲートウェイ	DHCP にて IP を動的取得する場合には設定は適用されません。
DNS サーバー	DHCP にて IP を動的取得する場合には設定は適用されません。設定する場合、最低 1 つ必須となります。2 つ以上の設定を推奨します。

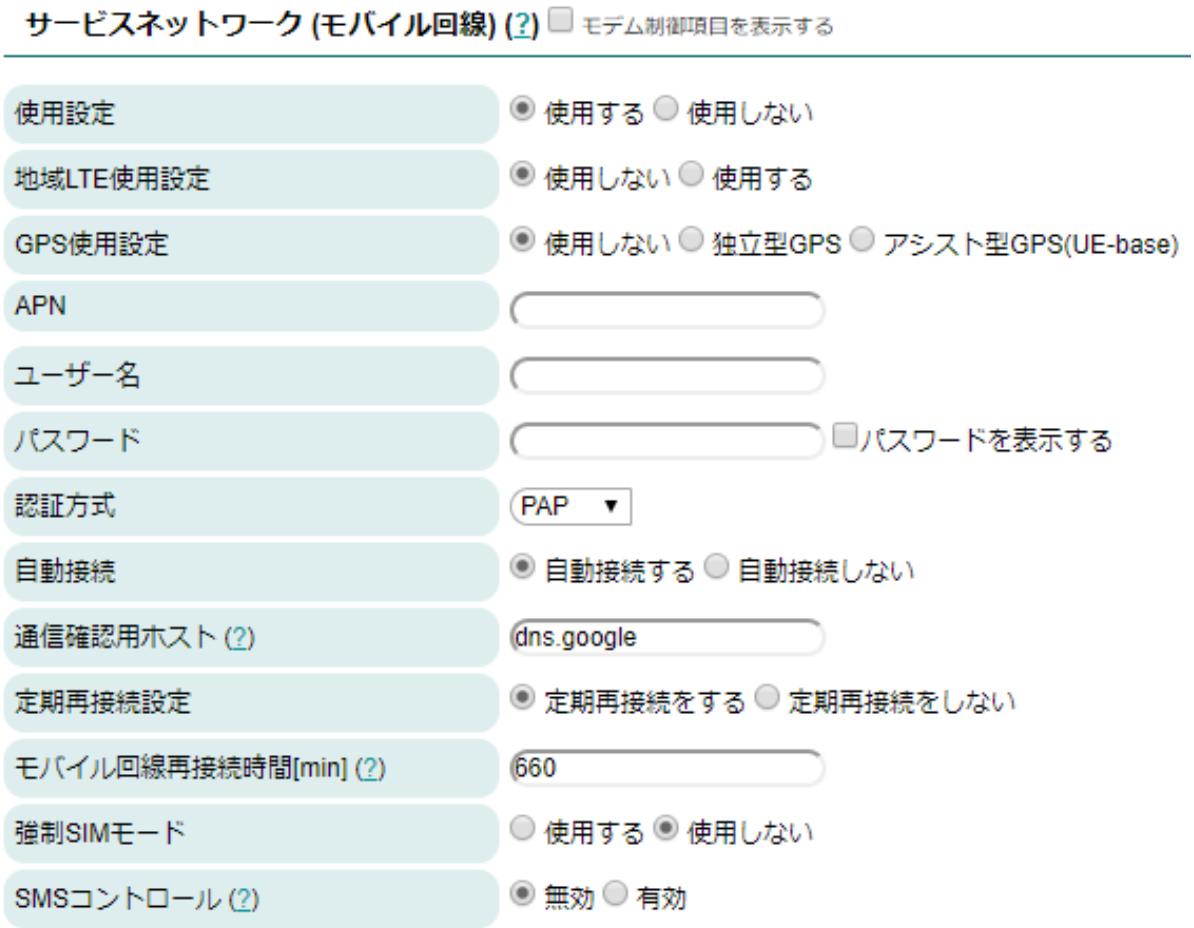
次の項から **モバイルルーター構成** と **サーバ構成** で設定方法が異なります。設定画面は上図と同じで、その下側の設定項目の解説となります。

3.3.1 モバイルルーター構成

本項では、本装置をモバイルルーターとして利用する際の設定方法を解説します。

サービスネットワーク (wlanN)	
項目	備考
使用設定	「使用する」を選択。
使用モード	「AP モード」を選択。
使用周波数	「2.4GHz」か「5GHz」を選択。
SSID	任意のアクセスポイント名を入力。SSID を一般から見えないようにするには、ステルス SSID フラグにチェックを入れます。
無線認証と無線暗号化	プルダウンメニューから任意のモードを選びます。デフォルトの設定のままで使用して問題ありません。
パスフレーズ (セキュリティキー)	8 文字以上を設定する必要があります。
AP 隔離機能	AP として起動した際に、クライアント間同士での通信を無効とする機能です。
802.11n 使用設定	AP として使用する場合、802.11n を用いた AP とするかの設定を行います。
IP アドレス	本装置の WLAN 向けの IP アドレスとネットマスクのビット数を入力します。
IP 配布レンジ	本設定では、DHCP サーバとして動作する為、配布する IP アドレス配布を設定します。
DHCP 用デフォルトゲートウェイ DHCP 用 DNS サーバ	DHCP クライアントに通知するデフォルトゲートウェイと DNS の IP アドレスを設定します。
固定 IP 設定	固定 IP を配布する際に使用する及び設定を行います。

サービスネットワーク (ethN)	
項目	備考
使用設定	使用する場合のみ、「使用する」を選択してください。
IP アドレス設定	Ethernet に設定する IP アドレスを設定します。静的を選択した場合、以下の項目が表示されます。
IP アドレス(静的)	静的アドレスを使用する場合には、本項目欄にて IP アドレスを設定してください。
DHCP 機能	サービスネットワーク (Wireless LAN) と同様に DHCP 機能を使用する場合に「使用する」を選択します。設定項目は同様に「DHCP 用デフォルトゲートウェイ」、「DHCP 用 DNS サーバ」、「固定 IP 設定」となります。



サービスネットワーク (モバイル回線)	
「モデム制御項目を表示する」にチェックは不要です。	
項目	備考
使用設定	「使用する」を選択してください。
地域 LTE 使用設定	地域 LTE 網 (Band 41) を使用する場合のみ、「使用する」を選択してください。 ※ LTE モジュール (NTT ドコモ/KDDI) または BWA モジュールを用いている場合のみ表示されます。
GPS 使用設定	GPS 機能を使用しない場合には「使用しない」を選択して下さい。また、使用する場合には下記の 2 種類から選択してください。「独立型 GPS」: 通信モジュールが GPS 衛星を補足し、本製品の位置情報を取得。「アシスト型 GPS(UE-base)」: 通信モジュールが GPS 衛星を補足しキャリア基地局情報と連動し、本製品の位置情報を計算。※ LTE モジュール (NTT ドコモ/KDDI) または BWA モジュールを用いている場合のみ表示されます。また、本機能はデータ収集時に使用する事が可能となります。尚、SIM を挿している必要があります。
APN ※ LTE モジュール (KDDI) の場合、項目はありません。	キャリア指定の APN を設定。
ユーザ名	キャリア指定のユーザ名を設定。
パスワード	キャリア指定のパスワードを設定。
認証方式	キャリア指定の認証方式を設定。
自動接続	「自動接続する」を選択すると、起動時から自動でモバイル回線へ接続します。
通信確認用ホスト	モバイル回線がインターネット等に接続されているかを検証するホストを指定します。尚、LTE モデムにおいて通信確認用ホストに対して一定回数確認が失敗すると再接続処理が行われます。 ※本項目が ”127.0.0.1 ”が設定されている場合、通信確認は行いません。そのため、再接続処理は実施されません。
定期再接続設定	モバイル回線を定期的に再接続するかの設定です。
モバイル回線再接続時間 [min]	モバイル回線接続後に本項目で設定した時間経過後に自動で切断及び接続を行います。
強制 SIM モード	モデム搭載時において、モデムデバイスファイル及び SIM カードが見えない場合、OpenBlocks 本体の再起動を行う設定です。
SMS コントロール	ここでは「無効」を設定。

以上、一連の設定が完了したら保存ボタンを押します。保存ボタンを押すと設定が保存され、ネットワーク設定については再起動後に適用されますので、**内部時計設定** 項に進んでください。

地域 LTE は地域 BWA と同一です。地域広帯域移動無線アクセス（地域 BWA：Broadband Wireless Access）システムは、2.5GHz 帯の周波数の電波を使用する無線システムです。

LTE モジュール (NTT ドコモ/KDDI) または BWA モジュール搭載の VX 及び EX シリーズにおいて、GPS を使用する場合には、GPS アンテナが必要となります。GPS アンテナご所望の方は、弊社の営業にご連絡ください。

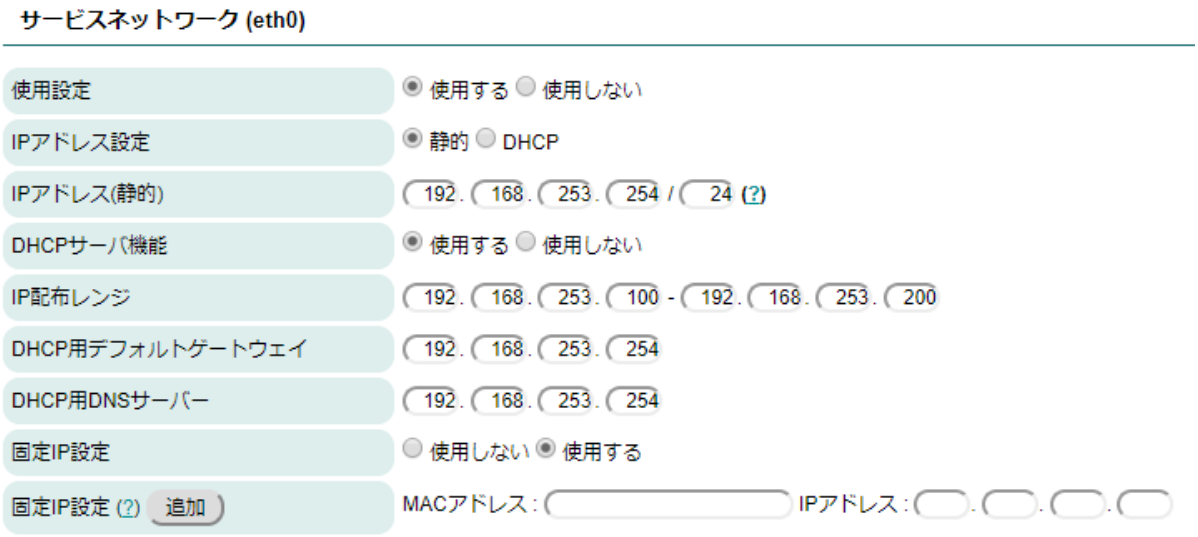
強制 SIM モードを有効にした場合、起動時にモデムのデバイスファイルが存在しないまたは SIM カードが認識できない場合には、5 分後に本体再起動が発生します。強制 SIM モードを有効にし、誤って SIM カードを抜いてしまって起動した場合等には、5 分以内に WEB UI にアクセスし強制 SIM モードを解除し再起動してください。

注釈: AP モードについては、管理用としての動作を前提としています。そのため、複数のクライアントによる接続性や送受信速度については実運用を想定しておりません。

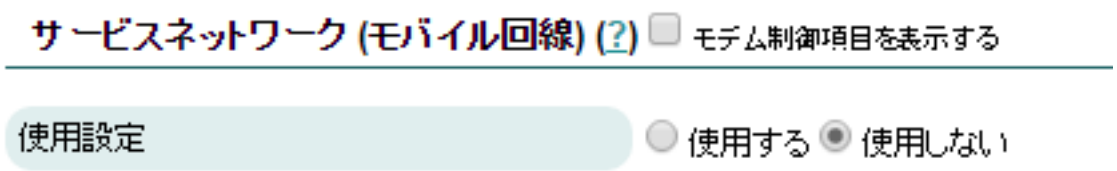
3.3.2 サーバ構成

本項では、本装置をネットワーク内の単体サーバとして利用する際の設定方法を解説します。

サービスネットワーク (wlanN)	
項目	備考
使用設定	「使用する」を選択。
使用モード	「クライアントモード」を選択。
SSID	接続するアクセスポイントの SSID を入力。ステルス SSID に対して接続する時はステルス SSID フラグをチェック。
無線認証	PSK(Pre Shared Key) か Enterprise から選択します。一般的な AP へのアクセスを行う場合には、PSK を指定してください。
IP アドレス設定	静的か DHCP を選択。DHCP の場合、本装置に DHCP サーバが固定 IP を配布するように設定してください。
IP アドレス(静的)	IP アドレスの設定が静的の時、IP アドレスを入力。
WLAN 検証用アドレス	WLAN の接続状態を監視するための ping を送出するサーバの IP または FQDN を入力。WLAN 上流の ping 応答可能な装置を設定します。



サービスネットワーク (ethN)
備考
使用する場合のみ、使用設定にて「使用する」を選択してください。また、静的アドレスを使用する場合には、IP アドレスを設定してください。 DHCP 機能を使用する場合には各項目のお設定が必要となります。



サービスネットワーク (モバイル回線)	
「モデム制御項目を表示する」にチェックは不要です。	
項目	備考
使用設定	「使用しない」を選択してください。

※「モデム制御項目を表示する」項目については、開発者向けの機能です。そのため、開発者向けガイドを確認してください。

以上、必要な項目を設定したら保存ボタンを押し、 **内部時計設定** 項に進んでください。

注釈: この項で存在しない上流アクセスポイントの SSID を登録してしまった場合、一般的な方法で本装置へのアクセスが出来なくなります。

この場合は、本装置を初期状態にして再起動する方法があります。

※ブラウザに WEB UI のセッション情報が残っている場合、以前の状態で残ったまま表示されます。そのため、ログアウトを行い再アクセスすることで使用許諾画面から再度設定してください。

◆ OpenBlocks IoT VX シリーズの場合

1, 本製品に USB コンソールを接続し PC と接続します。

2, 先ず本装置のパワースイッチを押して、本装置をシャットダウンします。

3, シャットダウン後にパワースイッチを押します。

4, GRUB メニューにて ”WebUI init boot ”を選択します。

5, 本装置のネットワークが工場出荷状態で起動してきます。

6, もう一度、本装置を設定し直し再起動します。

◆ OpenBlocks IoT BX 及び EX シリーズの場合

1, 先ず本装置のパワースイッチを押して、本装置をシャットダウンします。

2, 本装置の FUNC スイッチ (INIT スイッチ) を押しながらパワースイッチを押します。

ステータスインジケーターが一瞬点滅したらパワースイッチを離します。

ステータスインジケーターが黄色点灯したら FUNC スイッチを離します。

3, 本装置のネットワークが工場出荷状態で起動してきます。

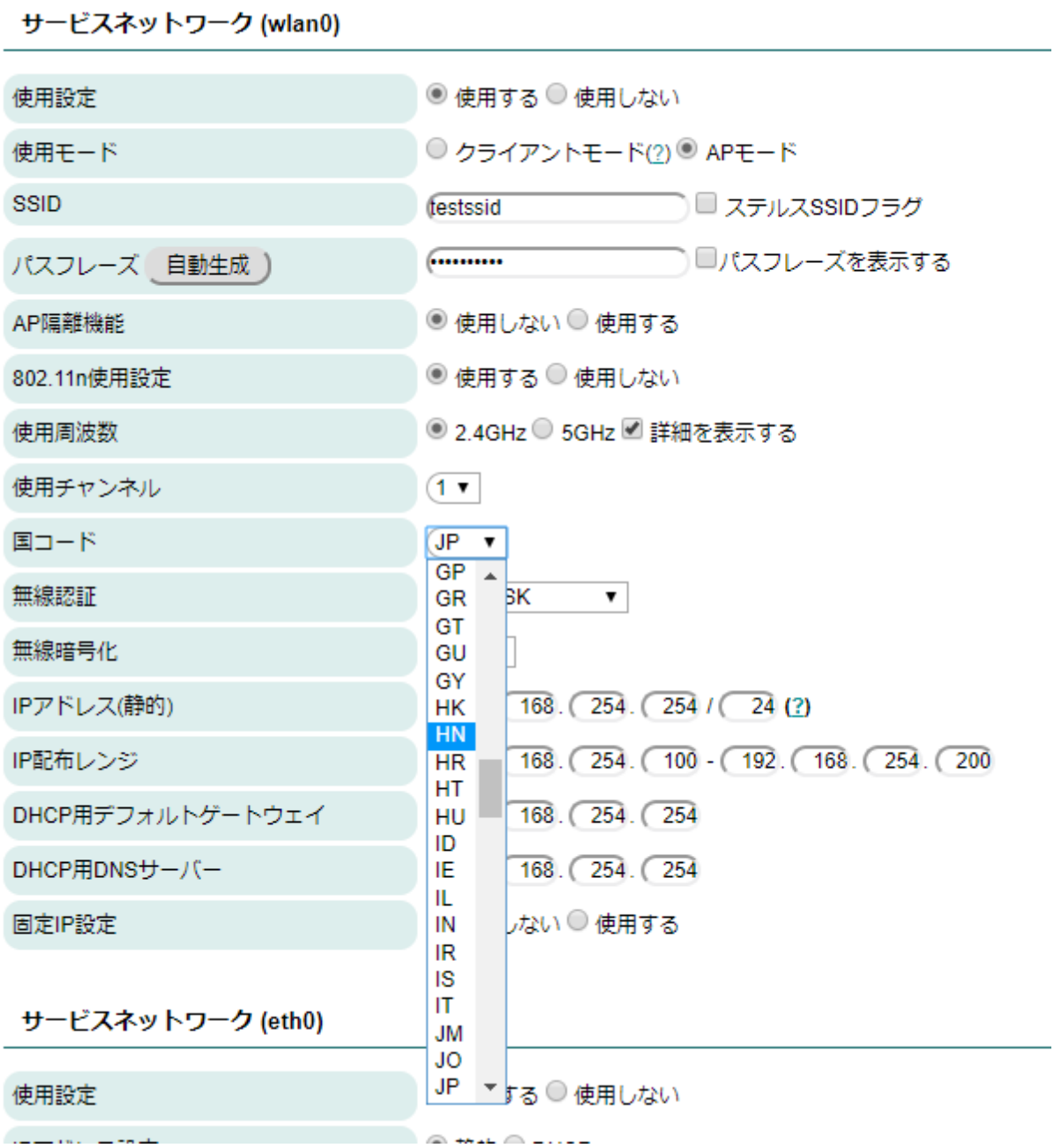
4, もう一度、本装置を設定し直し再起動します。

尚、ネットワークが工場出荷状態で起動後、設定せず再起動した場合、以前の設定のまま稼働します。

3.3.3 WLAN AP モードの詳細設定 (CH 設定と国際対応)

電波干渉によるチャネル変更や、日本国外での WLAN の AP モード利用における国コード設定が行えます。

サービスネットワーク (wlanN)	
項目	備考
使　用 設定	「AP モード」を選択。 「AP モード」を選択すると、使用周波数の右に「詳細を表示する」というチェックボックスが表示されます。 このチェックボックスにチェックを入れると、「使用チャネル」と「国コード」の設定項目が現れます。



サービスネットワーク (wlanN)	
項目	備考
使用チャネル	任意のチャネルをプルダウンメニューから選択します。空いているチャネルを見つけるにはスマートフォンの WLAN チャネルアナライザなどのアプリを使うと参考になります。尚、使用チャネルは 802.11n 使用設定にも依存します。使用可能なチャネルを事前にご確認ください。
国コード	本装置を設置する国に対応する国コードを設定してください。 日本の場合は「JP」となります。

3.3.4 Enterprise 認証について

AP モードにおいて WPA-Enterprise または WPA2-Enterprise 認証を行う場合、RADIUS サーバーと通信を行います。そのため、通信先の RADIUS サーバー及び通信インターフェースの設定を行います。

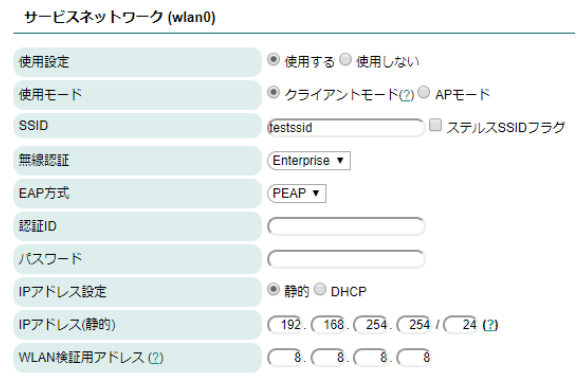
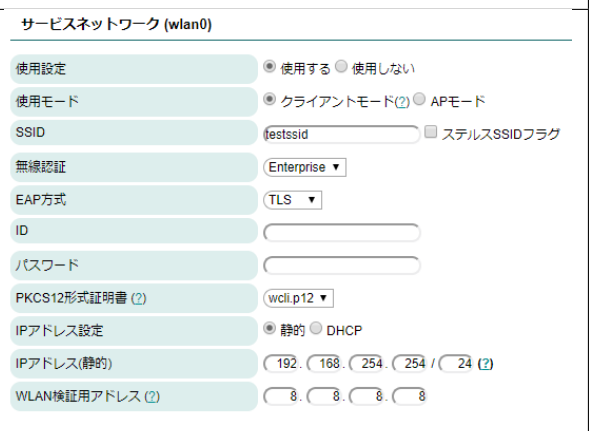
尚、OpenBlocks IoT BX/EX シリーズでは AP モードでの WPA-Enterprise または WPA2-Enterprise 認証については対応しておりません。

サービスネットワーク (wlan0)

使用設定	◉ 使用する ○ 使用しない
使用モード	○ クライアントモード(?) ◉ APモード
SSID	testssid ☐ ステルスSSIDフラグ
AP隔離機能	◉ 使用しない ○ 使用する
802.11n使用設定	◉ 使用する ○ 使用しない
使用周波数	◉ 2.4GHz ○ 5GHz ☐ 詳細を表示する
無線認証	WPA-Enterprise ▼
無線暗号化	AES ▼
認証サーバーアドレス	. . .
認証サーバーポート	1812
認証共有秘密鍵	
アカウントサーバーアドレス	. . .
アカウントサーバーポート	1813
アカウント共有秘密鍵	
認証通信インターフェース	eth0 ▼
IPアドレス(静的)	192 . 168 . 254 . 254 / 24 (?)
IP配布レンジ	192 . 168 . 254 . 100 - 192 . 168 . 254 . 200
DHCP用デフォルトゲートウェイ	192 . 168 . 254 . 254
DHCP用DNSサーバー	192 . 168 . 254 . 254
固定IP設定	◉ 使用しない ○ 使用する

サービスネットワーク (wlanN)	
項目	備考
認証サーバーアドレス	認証サーバーの IP アドレスを指定します。
認証サーバーポート	認証サーバーに接続する際のポートを指定します。 ※通常はデフォルトの 1812 から変更する必要はありません。
認証共有秘密鍵	認証サーバーと通信を行う際の認証共有秘密鍵を設定します。
アカウントサーバーアドレス	アカウンティングサーバーの IP アドレスを指定します。
アカウントサーバーポート	アカウンティングサーバーに接続する際のポートを指定します。 ※通常はデフォルトの 1813 から変更する必要はありません。
アカウント共有秘密鍵	アカウンティングサーバーと通信を行う際の認証共有秘密鍵を設定します。
認証通信インターフェース	認証及びアカウンティングサーバーにアクセスするインターフェースを選択します。

クライアントモードにおいて WPA-Enterprise または WPA2-Enterprise 認証を行う場合、追加のパラメーターを設定する必要があります。また、サポートしている EAP 方式は PEA と TLS となります。

PEAP 方式の場合	TLS 方式の場合
	

サービスネットワーク (wlanN)	
項目	備考
EAP 方式	PEAP と TLS から選択してください。
認証 ID(PEAP 時)	Enterprse 接続時に用いる認証用の ID を設定します。
パスワード (PEAP 時)	Enterprse 接続時に用いる認証時のパスワードを設定します。
ID(TLS 時)	Enterprse 接続時に用いる認証用の ID を設定します。
パスワード (TLS 時)	PKCS12 形式証明書を展開する際に用いるパスワードを指定します。
PKCS12 形式証明書 (TLS 時)	認証時に用いる証明書を選択します。 認証用の証明書は「ネットワーク」→「WLAN 証明書」タブからアップロードを行ってください。尚、証明書が存在しない場合、存在しない旨のメッセージが表示されます。

3.4 内部時計設定

本製品は RTC のバックアップ電池を搭載しております。しかし、基本的には NTP サーバとの時刻同期を推奨します。

但し、NTP サーバが利用できない環境での運用の場合には、本装置の WEB UI を表示している PC やスマートフォンの時刻を WEB ブラウザ上で同期できます。

時刻設定	
項目	備考
PC と時刻を同期	同期ボタンを押すと WEB を表示している PC の時刻を反映します。
タイムゾーン	本装置の設置地域を選択します。
時刻同期設定	時刻同期の方式を設定します。通常は NTP または定期 NTP を指定してください。NTP は NTP サーバと常時通信を行い、時刻補正を行います。定期 NTP は定期的に NTP サーバと時刻同期を行い、時刻同期を行います。 LTE モジュール (NTT ドコモ) を搭載している場合、"モデム"項目が表示されモデムから時刻同期を行うことが可能です。(SIM が挿入されている必要があり、また正しい APN の設定が必要になります。)
NTP サーバ (NTP 選択時)	時刻同期の方式を設定します。通常は NTP または定期 NTP を指定してください。

位置情報設定	
項目	備考
位置情報同期	同期ボタンを押すと WEB を表示している PC の時刻を反映します。
緯度	本装置の設置地域を選択します。
経度	時刻同期の方式を設定します。通常は NTP または定期 NTP を指定してください。

編集後、保存ボタンを押すと設定が保存されます。基本的には再起動は不要ですが、使用しているアプリケーションのタイムゾーン情報等の反映があるため、再起動を推奨します。

ここまでが本装置を運用するために必要な基本的な設定項目です。

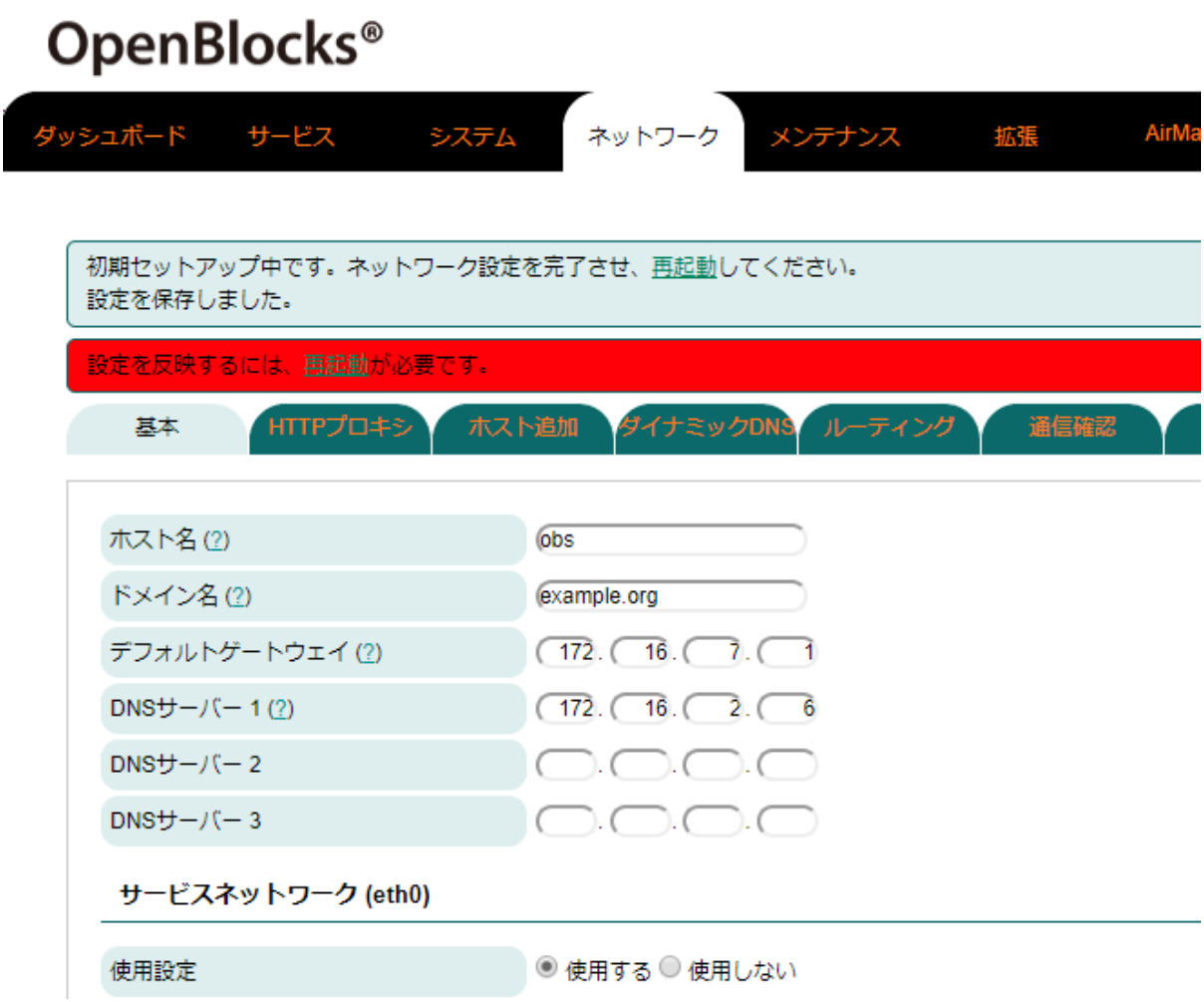
設定が完了後に、次項のシステム再起動を実施します。

3.5 システム再起動による設定項目の反映

ここまでが本装置を運用するために必要な最小限の設定項目です。

その他の設定項目については必要に応じて解説部分を参照してください。

本項ではネットワークの基本設定後、システムに設定内容を反映するためのシステム再起動について解説を進めます。



ネットワークの基本設定後、保存ボタンを押した状態になると WEB 画面の上部にシステム再起動を促すメッセージが左図の通り表示されます。

システム再起動には、この赤枠で表示されたメッセージの「再起動」リンクをクリックします。クリックするとメンテナンスメニュー内の停止、再起動タブに表示が切り替わります。

この画面内の再起動の実行ボタンを押します。

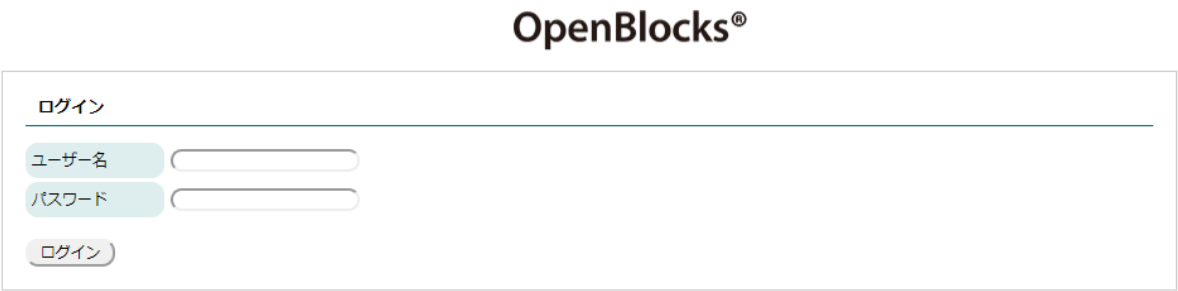
更に再起動の確認画面が現れるので、実行ボタンを押すと、最終確認ウィンドウがポップアップします。

これが最後の確認で「OK」ボタンを押すとシステム再起動が始まります。

再起動はシステムの状態によりますが、表示されている秒数程度お待ちください。

無線経由で WEB UI にアクセスし、本装置が AP モードの場合、再起動後に本装置への再接続が発生します。また、再起動完了後にログイン画面を表示させるには WEB ブラウザからのリロード操作が必要です。

3.6 管理者ログイン画面



本装置が出荷直後の状態にない時、最初に表示される画面です。

一度ログアウトしてしまっても、この画面からのスタートになるので、その場合は、ここでログインしてください。

3.7 ダッシュボード画面

本装置の WEB UI にログインすると最初に表示される画面です。

ここでは OpenBlocks IoT Family のハードウェアリソースやネットワーク情報等を表示します。

最新の情報を表示させるには更新ボタンを押してください。

第4章　SMSコントロール

本装置は一部のモバイル回線モデムモジュールにて SMS をサポートしています。

(モバイル回線契約に SMS 機能が無い場合、サポートできません。)

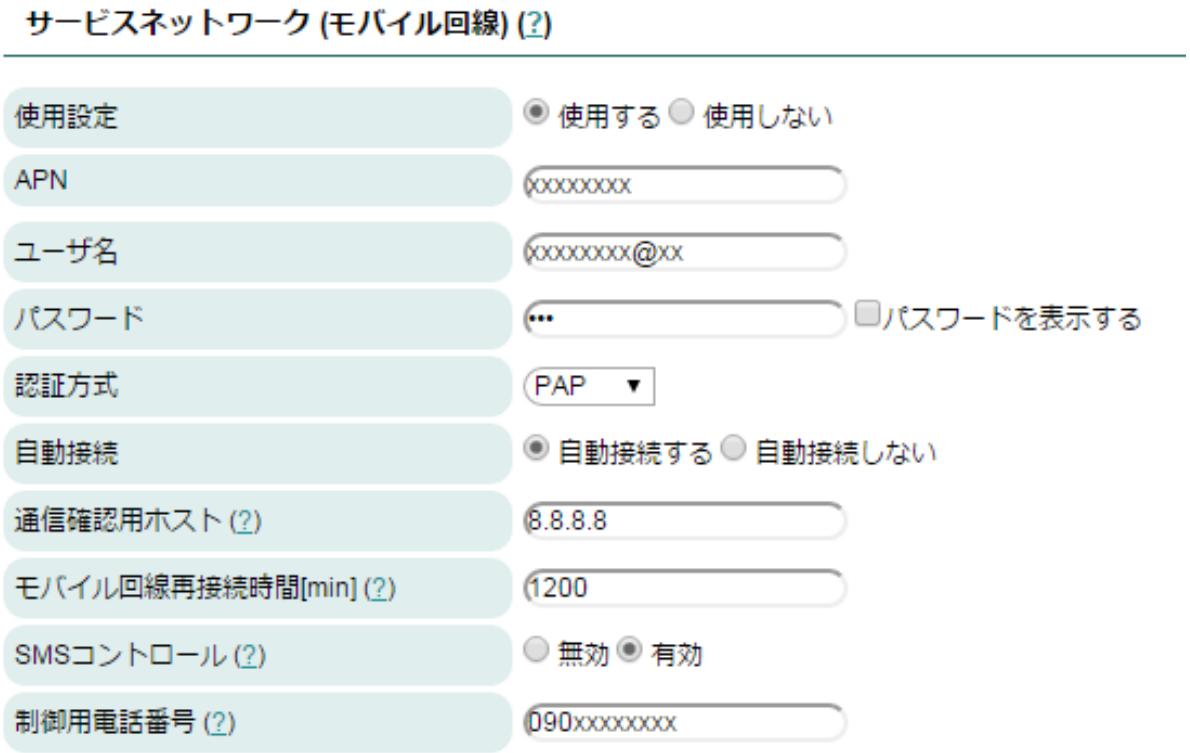
SMS とは、携帯電話で使えるショートメッセージサービスで、最大約 70 文字前後のメッセージを相手の電話番号向けて送信する機能です。本装置が通常使用しているデータ通信とは異なります。

本装置では、特定のキーワードの SMS を受信することによってデータ通信を開始・停止やシェルスクリプトの実行を行うことが出来ます。

※ KDDI キャリアの SIM の場合には利用できません。

4.1 SMS コントロールの起動設定

SMS コントロールはモバイル回線を使用されている方向けの機能です。モバイル回線の設定については *WEB UI* **の初期基本設定** の モバイルルーター構成 項を参照ください。



サービスネットワーク (モバイル回線)	
項目	備考
自動接続	この設定はどちらでも構いません。尚、SMS コントロールにてモバイル回線を接続した場合及び網側から回線切断された場合には、再接続は行われません。
SMS コントロール	「有効」を設定。
制御用電話番号	SMS コントロールを「有効」に設定すると、表示される項目です。ここには SMS 制御をするスマホ等の電話番号を入力します。ここに設定した電話番号以外からの SMS は無視されます。市街局番からの電話番号を入力します。 尚、プライベート回線用の SMS では 4 桁等の短い場合があります。尚、","区切りにて複数の制御用電話番号を登録することができます。

4.2 SMS コントロールのコマンド

SMS コントロールには以下のコマンドが組み込まれています。

コマンド	コマンド内容	備考
CON	モバイル回線を接続する	
COFF	モバイル回線を切断する	
SSHON	SSH を開放する	SSH 解放後に OS を再起動すると自動的に閉鎖されます。再起動までは SSH 解放状態となるため、利用後は閉鎖してください。
SSHOFF	SSH を閉鎖する	
REBOOT	システムを再起動する	
USCR1 ～ USCR5	ユーザースクリプトをバックグラウンド実行	WEB UI の拡張タブにあるスクリプトエディタで編集可能です。登録方法については「*SMS* **ユーザ定義スクリプトの登録**」を参照してください。
USCR1F ～ USCR5F	ユーザースクリプトをフォアグラウンド実行	
UP-GRADE	オンラインのアップデート処理を実行します	
STUN-NEL	SSH トンネルを構築	

4.3 SMS での複数コマンド送信

1 回の SMS で複数のコマンドを一括で送信可能です。

"CON", "COFF", "SSHON", "SSHOFF", "USCR1F"〜"USCR5F", "UPGRADE"はフォアグラウンドで実行されるので、SMS の送信文字列でたとえば以下のように ”+でつなぐと順次実行されます。

例)

CON+USCR1F+USCR2F+COFF		モバイル回線を接続、スクリプト 1 実行、スクリプト 2 実行、モバイル回線を切断。
CON+SSHON	:	モバイル回線を接続してから SSH を開放します。
SSHOFF+COF	:	SSH を閉鎖してからモバイル回線を切断します。

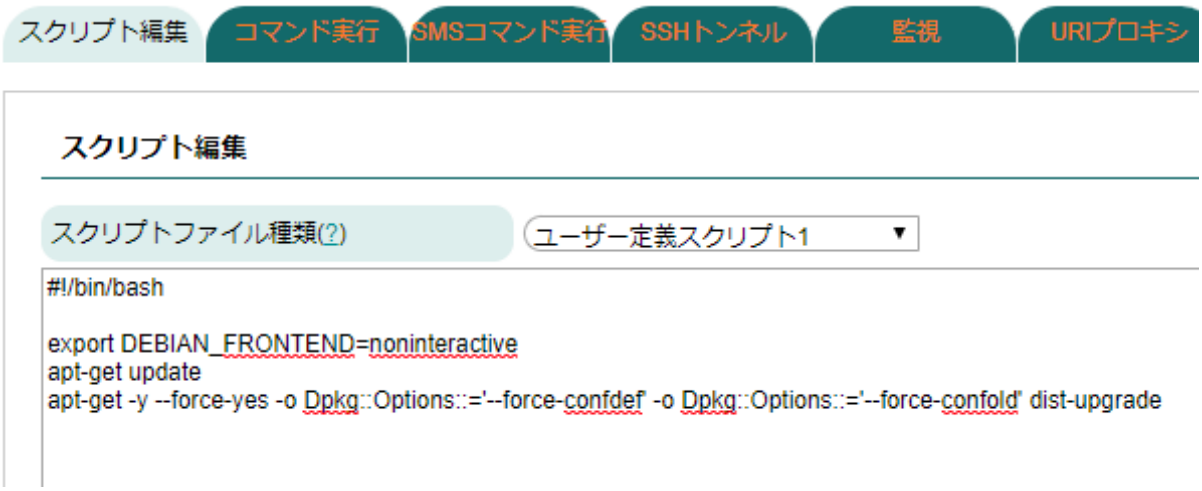
※"USCR1" 〜 "USCR5"及び"STUNNEL"はバックグラウンド実行になるため、並列処理になります。

4.4 SMS ユーザ定義スクリプトの登録

ユーザが定義したスクリプトを WEB UI にて登録・編集が出来ます。尚、本機能は Linux のシェルスクリプトをご自身で作成できる方向けの機能です。スクリプトの実施内容については弊社サポート対象外となります。

スクリプト作成及び編集は「拡張」タブ内にあるスクリプト編集にて行います。

スクリプト編集
スクリプトの種類
プルダウンメニューから編集するスクリプトを選んでください。メニュー中にある「起動スクリプト」には本装置の OS 起動時に自動実行させるスクリプトを記述することが出来ます。 尚、起動スクリプトに記載されたスクリプトはバックグラウンドで実行されます。



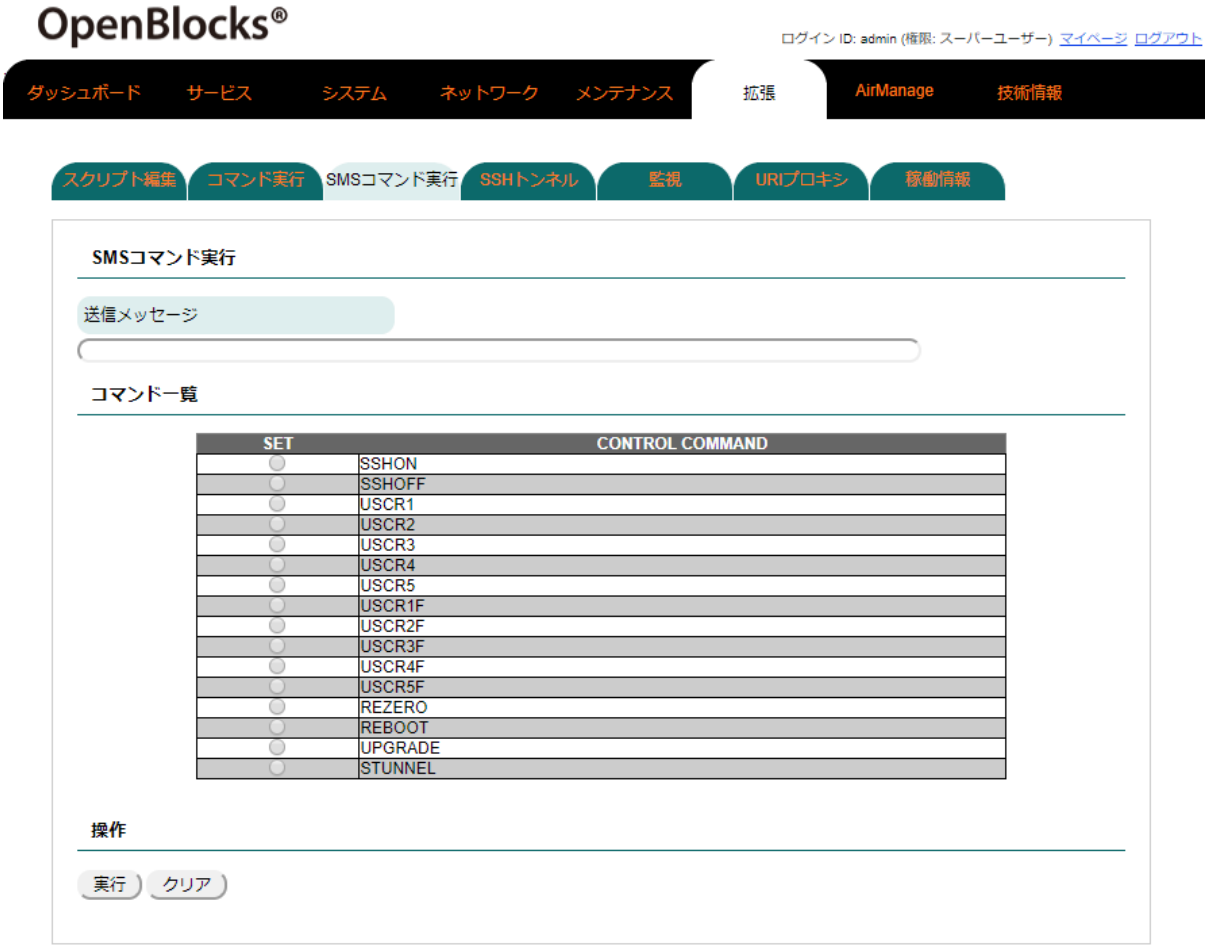
この欄にスクリプトを記述します。このスクリプト例では各アプリケーションのアップデートが行えます。但し、インターネット環境内です。(各アプリケーションのセキュリティアップデートは頻繁に行われる為、おすすめのスクリプトです。)

スクリプトが完成したら画面左下側にある保存ボタンを押してください。 また、不要なスクリプトは削除ボタンにて消去できます。

※上記の参考例では、遠隔地にある本装置に対して SMS 経由による OS パッチを当てる内容となっております。

4.5 SMS コントロールコマンドのダイレクト実行

本装置に登録された SMS コントロールコマンドは通常携帯電話で命令を発行し実行させせますが、WEB UI からも直接実行させることが出来ます。



SMS コマンド実行	
項目	備考
送信メッセージ	ここへ疑似的に送信する SMS コマンドを入力します。
コマンド一覧	SMS コマンドの一覧の SET 部を選択すると送信メッセージに対象のコマンドが追加されます。 2 個目以降については自動で"+"が挿入されます。 ※"CON"及び"COFF"はモバイル回線を「使用する」に設定している場合にのみ表示されます。
保存	送信メッセージに入力されたコマンドを本装置に疑似送信します。
クリア	送信メッセージの中身を消去します。

第5章　サービス機能

本装置の標準状態のサービス機能では、基本機能として BT インターフェースの制御及び各種デバイスの登録機能のみサポートします。尚、OpenBlocks IX9 では BT インターフェースデバイスを接続した場合にのみ表示されます。

通常では、サービスタブを選択すると以下のような画面が表示されます。

基本機能のリンクを押すと以下のような画面に遷移します。

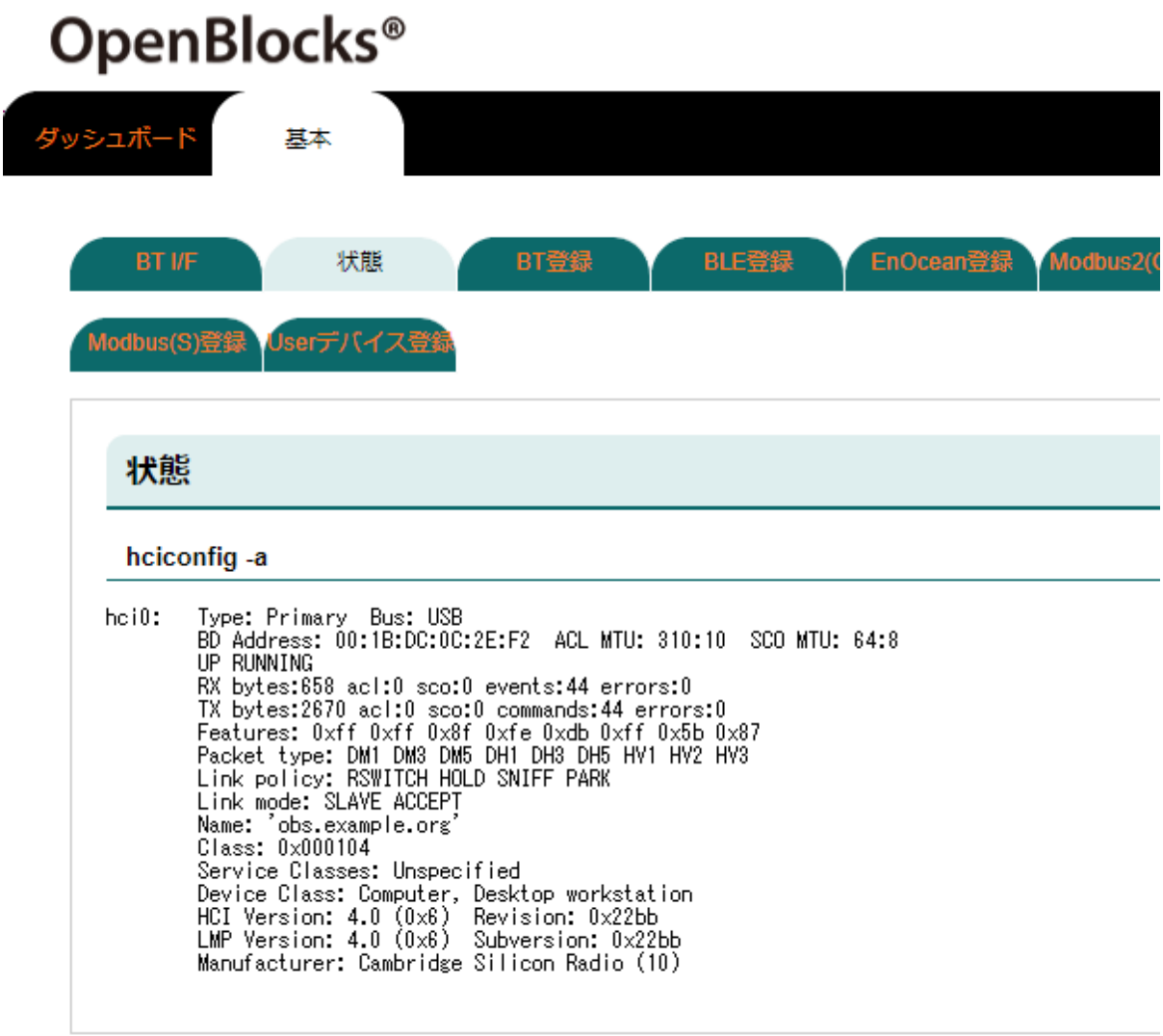
5.1 BT I/F 制御

OpenBlocks シリーズが IoT デバイスとして標準サポートしているインターフェースの一つとして BT があります。BT のインターフェースの制御用として、『BT I/F』タブから設定が可能です。(OpenBlocks IX9 では BT インターフェースデバイスを接続している場合のみ、表示されます)

BT I/F	
項目	備考
hciN 使用設定	BT のインターフェースの使用設定が可能です。 『使用する』にした場合、BT I/F が UP 状態となります。 また、『使用しない』にした場合、BT I/F が DOWN 状態となります。

5.2 状態

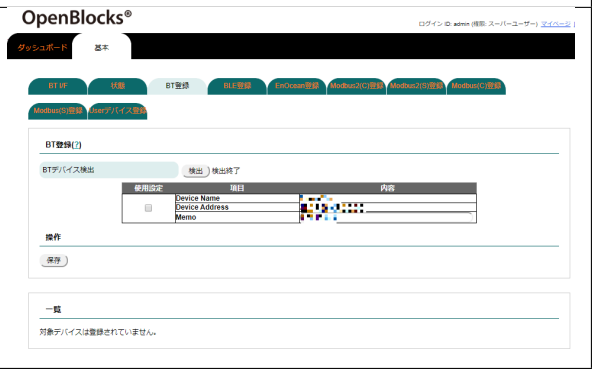
OpenBlocks シリーズが IoT デバイスとして標準サポートしているインターフェースの一つとして BT 状態を『状態』タブから確認できます。(OpenBlocks IX9 では BT インターフェースデバイスを接続している場合のみ、表示されます)



状態	
項目	備考
hciconfig -a	BT のインターフェースの状態を確認できます。

5.3 BT 登録

BT I/F が UP 状態となっている場合、BT デバイスの登録が行えます。

検出前	検出後
	

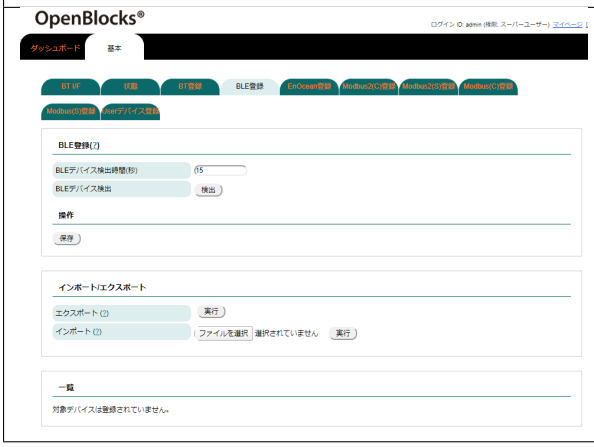
BT 登録	
項目	備考
BT デバイス検出	「検出」のボタンを押すと周囲に存在する BT デバイスを一覧に表示します。一覧の中から利用するデバイスの使用設定にチェックを入れることでペアリングが実行されます。ペアリング完了後に保存ボタンを押すことで登録されます。
Device Name	BT デバイスの検出の際に取得したディスカバリデータをもとにデバイス名を表示します。
Device Address	BT デバイスの検出の際に取得したディスカバリデータをもとにデバイスアドレスを表示します。
Memo	BT デバイスの検出の際に取得したディスカバリデータをもとにデバイス名をデフォルトで設定します。このフィールドは編集可能ですので修正が必要な場合には適宜修正を行ってください。

尚、BT デバイスが登録されている場合、一覧部に登録デバイスの一覧が表示されます。ここからデバイスの削除及びメモ情報の更新が行えます。

5.4 BLE 登録

BT I/F が UP 状態となっている場合、BLE デバイスの登録が行えます。

注釈: IoT データ制御機能等にて BT I/F を使用している場合、検出が行えません。そのため、事前に BT I/F を使用しているプロセスを停止させてください。尚、IoT データ制御機能が使用している場合には、ダッシュボードから停止可能です。

検出前	検出後
	

BLE 登録	
項目	備考
BLE デバイス検出時間(秒)	BLE デバイスの検出時間を秒単位で設定します。
BLE デバイス検出	「検出」のボタンを押すと周囲に存在する BLE デバイスを一覧に表示します。一覧の中から利用するデバイスの使用設定にチェックを入れ保存ボタンを押すことで登録されます。
Device Name	BLE デバイスの検出の際に取得したアドバタイズデータをもとにデバイス名を表示します。
Device Address	BLE デバイスの検出の際に取得したアドバタイズデータをもとにデバイスアドレスを表示します。
Memo	BLE デバイスの検出の際に取得したアドバタイズデータをもとにデバイス名をデフォルトで設定します。このフィールドは編集可能ですので修正が必要な場合には適宜修正を行ってください。

尚、BLE デバイスが登録されている場合、一覧部に登録デバイスの一覧が表示されます。ここからデバイスの削除及びメモ情報の更新が行えます。

インポート/エクスポート

インポート/エクスポート	
項目	備考
エクスポート	本筐体が保持している BLE デバイス情報を JSON ファイルとしてエクスポートします。
インポート	JSON ファイルを入力し、本筐体へ登録及び更新したい BLE デバイス情報をインポートします。
保存	インポートした BLE デバイスの JSON ファイルの内容を本筐体へ保存します。

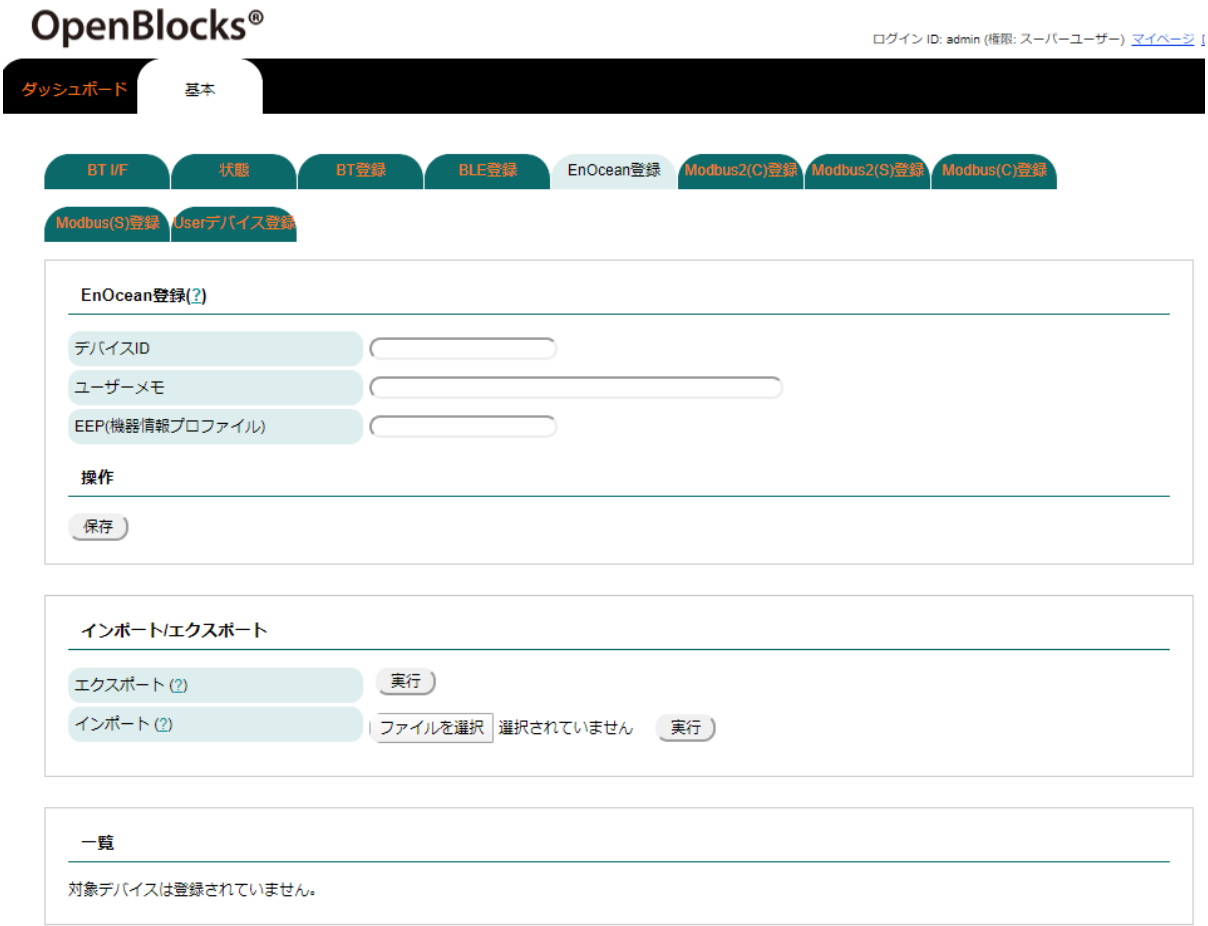
JSON ファイルの内容は WEB UI のバージョンによって異なる恐れがあります。

そのため、エクスポートした JSON ファイルを参考に作成してください。

5.5 EnOcean 登録

本装置に EnOcean 拡張モジュールを載せている場合、EnOcean でのデバイスの情報を取得できます。(拡張機能から IoT データ制御機能をインストールしている場合に限ります)

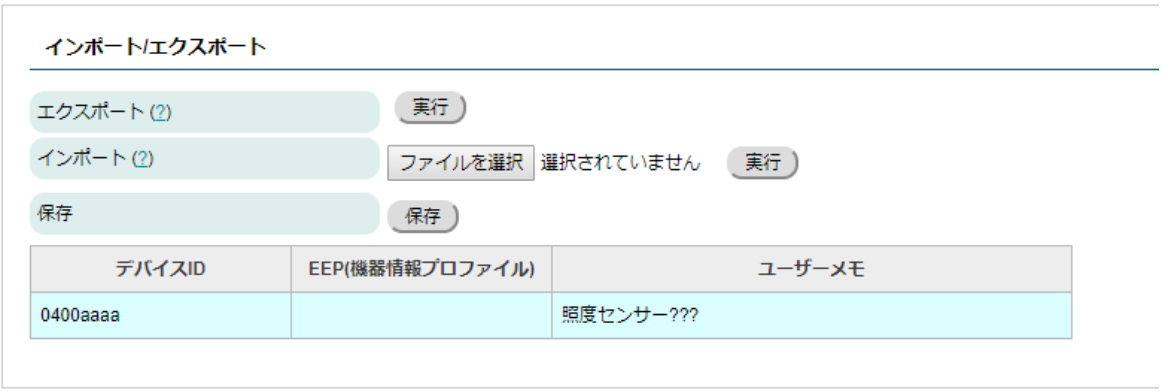
そのため、『EnOcean 登録』タブから取得対象とする EnOcean デバイスの登録が行えます。



EnOcean 登録	
項目	備考
デバイス ID	登録対象の EnOcean デバイスのデバイス ID を設定します。
ユーザーメモ	登録対象の EnOcean デバイスに対して、メモを設定することが行えます。のユーザーメモはクラウドとのデータ通信に用いることができます。
EEP(機器情報プロファイル)	登録対象の EnOcean デバイスの EEP を設定することができます。この EEP に正しい情報を設定している場合、EnOcean のデバイスデータを温度・湿度等の情報を制御することができます。

尚、EnOcean デバイスが登録されている場合、一覧部に登録デバイスの一覧が表示されます。ここからデバイスの削除及びメモ情報の更新が行えます。

インポート/エクスポート



インポート/エクスポート	
項目	備考
エクスポート	本筐体が保持している EnOcean デバイス情報を JSON ファイルとしてエクスポートします。
インポート	JSON ファイルを入力し、本筐体へ登録及び更新したい EnOcean デバイス情報をインポートします。
保存	インポートした EnOcean デバイスの JSON ファイルの内容を本筐体へ保存します。

JSON ファイルの内容は WEB UI のバージョンによって異なる恐れがあります。

そのため、エクスポートした JSON ファイルを参考に作成してください。

5.6 Modbus2(C) 登録

PD Handler Modbus2 Client にて Modbus プロトコルを話すデバイスの登録ができます。この登録したデバイス情報をもとに IoT データ制御機能にて送受信等の設定が可能になります。(IoT データ制御機能はメンテナンス → 拡張機能からインストールしてください)

そのため、『Modbus2(C) 登録』タブからデバイスの登録が行えます。

※ Modbus2 クライアントデバイスは本筐体から対象デバイスに対して、データを取得するデバイスとしています。

Modbus2 クライアントデバイス	
項目	備考
ユーザーメモ	登録対象の Modbus2 クライアントデバイスに対して、メモを設定することが行えます。このユーザーメモは内部的な情報を保存するフィールドとなります。そのため、クラウド等にデータは送信されません。

※本デバイスの登録ではメモのみの登録となります。使用するデバイスファイル等の設定に関しては本項目では設定は行いません。

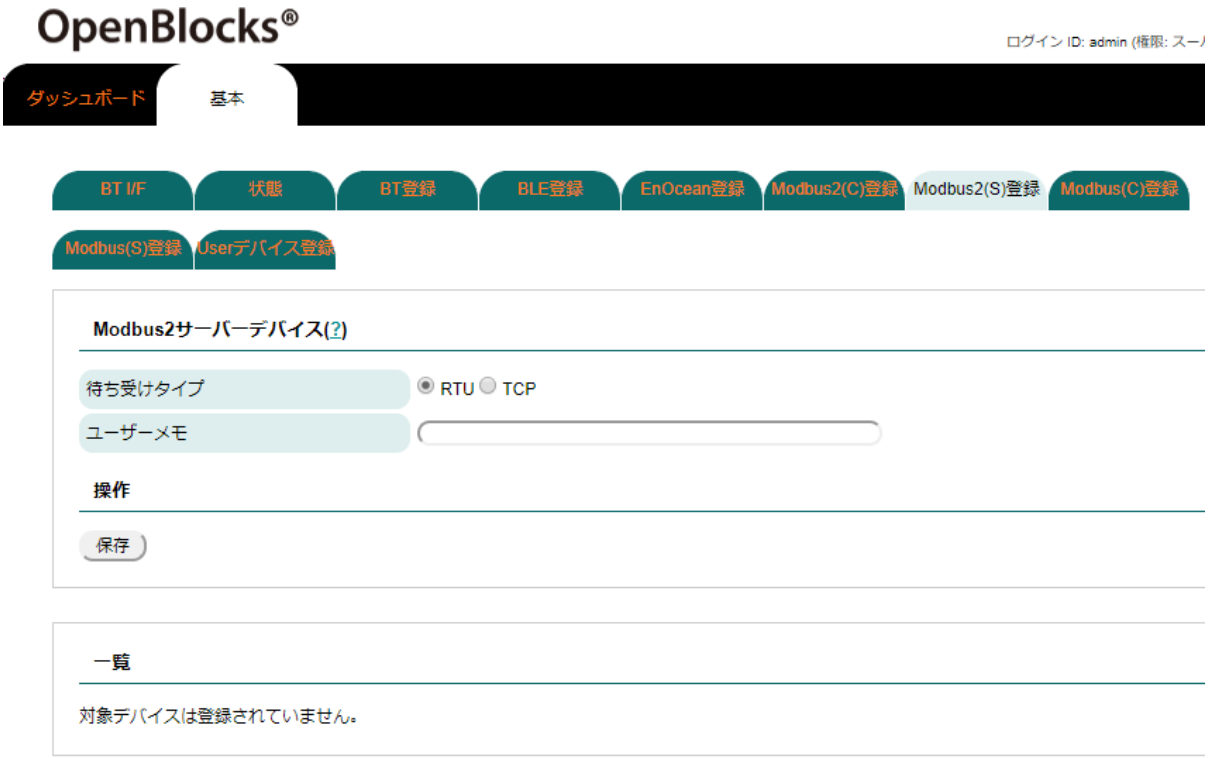
尚、Modbus2 クライアントデバイスが登録されている場合、一覧部に登録デバイスの一覧が表示されます。ここからデバイスの削除及びメモ情報の更新が行えます。

5.7 Modbus2(S) 登録

PD Handler Modbus2 Server にて Modbus プロトコルを話すデバイスの登録ができます。この登録したデバイス情報をもとに IoT データ制御機能にて送受信等の設定が可能になります。(IoT データ制御機能はメンテナンス → 拡張機能からインストールしてください)

そのため、『Modbus2(S) 登録』タブからデバイスの登録が行えます。

※ Modbus2 サーバーデバイスは本筐体に対して対象デバイスが Modbus プロトコルにて送信するデバイスとしています。



Modbus2 サーバーデバイス	
項目	備考
待ち受けタイプ	登録対象の Modbus サーバーデバイスに対して、待ち受けタイプを設定します。待ち受け方法は以下の 2 種類から設定可能です。 ・TCP : Ethernet 等のネットワークでの待ち受け ・RTU : シリアルデバイスファイルでの待ち受け
ユーザーメモ	登録対象の Modbus2 サーバーデバイスに対して、メモを設定することが行えます。このユーザーメモは内部的な情報を保存するフィールドとなります。そのため、クラウド等にデータは送信されません。

※本デバイスの登録では待ち受けタイプ・メモのみの登録となります。使用するデバイスファイル等の設定に関しては本項目では設定は行いません。

尚、Modbus2 サーバーデバイスが登録されている場合、一覧部に登録デバイスの一覧が表示されます。ここからデバイスの削除及びメモ情報の更新が行えます。

5.8 Modbus(C) 登録

Modbus プロトコルを話すデバイスの登録ができます。この登録したデバイス情報をもとに IoT データ制御機能にて送受信等の設定が可能になります。(IoT データ制御機能はメンテナンス → 拡張機能からインストールしてください)

そのため、『Modbus(C) 登録』タブからデバイスの登録が行えます。

※ Modbus クライアントデバイスは本筐体から対象デバイスに対して、データを取得するデバイスとしています。

Modbus クライアントデバイス	
項目	備考
ユーザーメモ	登録対象の Modbus クライアントデバイスに対して、メモを設定することが行えます。このユーザーメモはクラウドとのデータ通信に用いることがあります。

※本デバイスの登録ではメモのみの登録となります。使用するデバイスファイル等の設定に関しては本項目では設定は行いません。

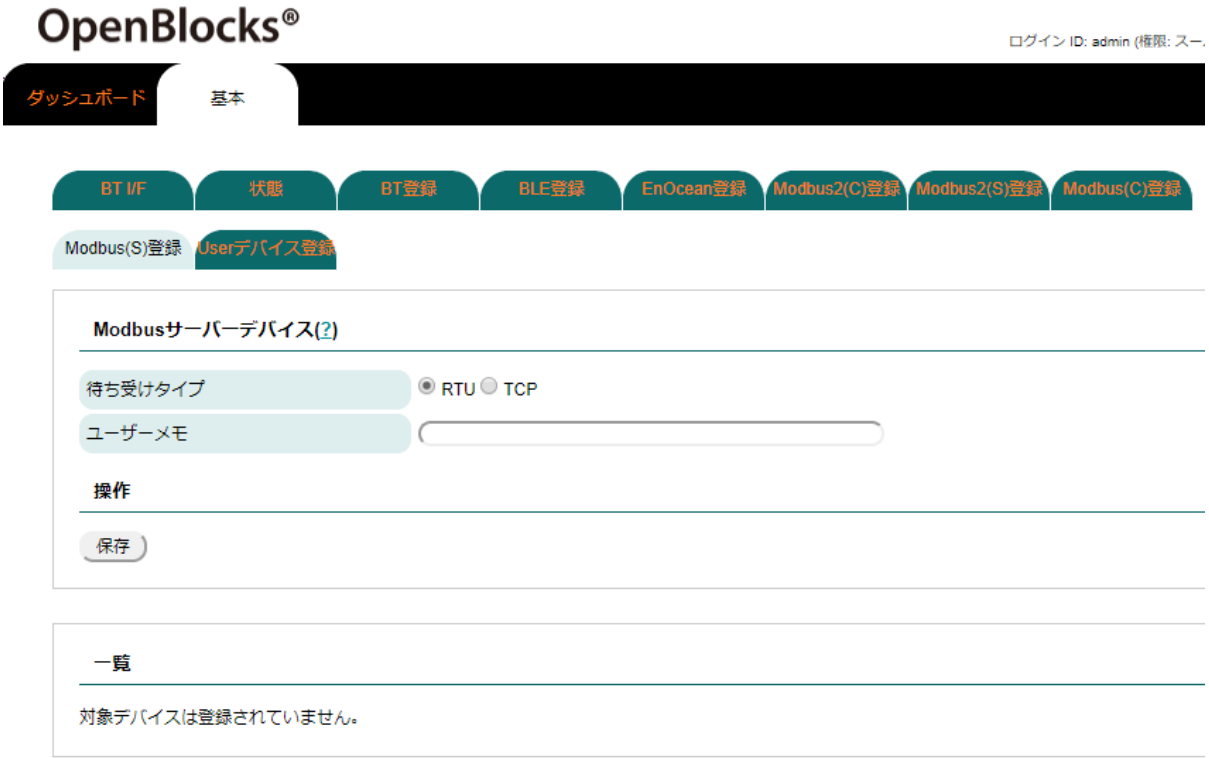
尚、Modbus クライアントデバイスが登録されている場合、一覧部に登録デバイスの一覧が表示されます。ここからデバイスの削除及びメモ情報の更新が行えます。

5.9 Modbus(S) 登録

Modbus プロトコルを話すデバイスの登録ができます。この登録したデバイス情報をもとに IoT データ制御機能にて送受信等の設定が可能になります。(IoT データ制御機能はメンテナンス → 拡張機能からインストールしてください)

そのため、『Modbus(S) 登録』タブからデバイスの登録が行えます。

※ Modbus サーバーデバイスは本筐体に対して対象デバイスが Modbus プロトコルにて送信するデバイスとしています。



Modbus サーバーデバイス	
項目	備考
待ち受けタイプ	登録対象の Modbus サーバーデバイスに対して、待ち受けタイプを設定します。待ち受け方法は以下の 2 種類から設定可能です。 ・TCP : Ethernet 等のネットワークでの待ち受け ・RTU : シリアルデバイスファイルでの待ち受け
ユーザーメモ	登録対象の Modbus サーバーデバイスに対して、メモを設定することが行えます。このユーザーメモはクラウドとのデータ通信に用いることがあります。

※本デバイスの登録では待ち受けタイプ・メモのみの登録となります。使用するデバイスファイル等の設定に関しては本項目では設定は行いません。

尚、Modbus サーバーデバイスが登録されている場合、一覧部に登録デバイスの一覧が表示されます。ここからデバイスの削除及びメモ情報の更新が行えます。

5.10 User デバイス登録

上記のデバイス種類とは異なるデバイスを仮想的な登録ができます。この登録したデバイス情報をもとに IoT データ制御機能にて送受信等の設定が可能になります。(IoT データ制御機能はメンテナンス → 拡張機能からインストールしてください)

User デバイス	
項目	備考
ユーザーメモ	登録対象の User デバイスに対して、メモを設定することが行えます。

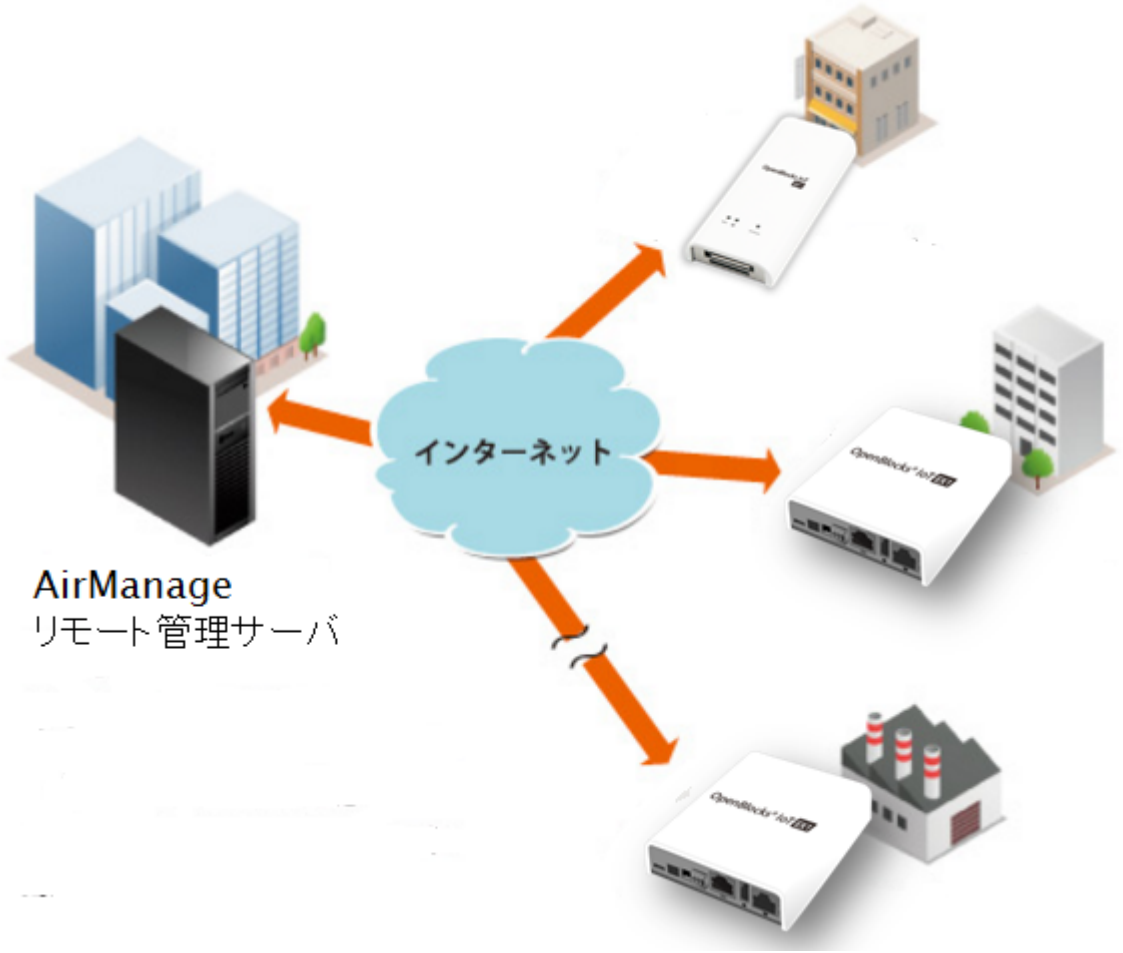
尚、User デバイスが登録されている場合、一覧部に登録デバイスの一覧が表示されます。ここからデバイスの削除及びメモ情報の更新が行えます。

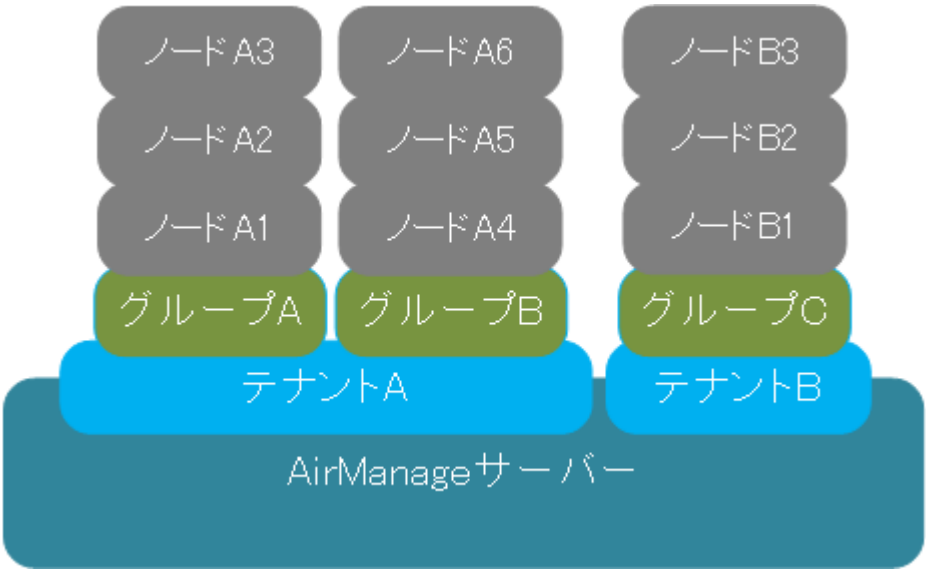
第6章　AirManage機能

AirManage は遠隔地に配備した OpenBlocks シリーズを管理する機能です。

AirManage はインターネット上に用意している AirManage リモート管理サーバと各 OpenBlocks 間で通信を行い、各コンフィグ管理等を行います。

また、AirManage の詳細機能やサービス加入等については、『**設定項目別リファレンス** の OpenBlocks のサービス及び技術情報一覧』をご確認ください。





注釈: AirManage を初めてご利用する場合には、利用規約 をご確認ください。

6.1 AirManage 初回アクセス設定

AirManage サービスを使用する場合には、事前に AirManage リモート管理サーバ側に OpenBlocks 個体を登録している必要があります。

登録後に各 OpenBlocks がサーバに初回アクセスすることでは AirManage サービスが使用可能となります。その為、AirManage アカウントの作成や AirManage サーバに OpenBlocks 登録が完了していない場合には次項の"AirManage 登録"を先にご確認ください。

初回アクセスする為の設定は「AirManage」→「AirManage」タブから適用を行います。

※本項での初回アクセスする際に用いるネットワークは「ネットワーク」→「基本」タブを引き継ぎます。そのため、インターネット環境へアクセスする為の準備を事前に設定してください。

※弊社、出荷時に AirManage キッティングオプションを適用している場合には不要となります。

AirManage	
項目	備考
使用設定	AirManage サービスに参加する場合、「使用する」を選択してください。また、サービスから解約する場合には「使用しない」を選択してください。
適用方法	以下から選択してください。 ●サービス加入のみ AirManage リモート管理サーバへアクセスを行うのみです。コンフィグは適用されませんが、サービスに加入し各種機能が使用可能となります。 ●ゼロコンフィグ (ネットワーク設定保持) AirManage リモート管理サーバからコンフィグをダウンロードし、ネットワークタブの設定以外の部分を適用します。 ●サービス加入のみ AirManage リモート管理サーバからコンフィグをダウンロードし適用します。
サービス適用URL	サービス加入の際に弊社から連絡のあった FQDN 情報をフォームに入力します。
事前確認	「確認」ボタンを押すことでノード側のネットワーク及び設定している URL 情報を用いて、AirManage サーバ側に登録されているか確認可能です。

事前確認の「確認」ボタンを押し AirManage が問題なく利用できる状態の場合、「保存＆実行」ボタンが

表示されます。「保存＆実行」を押した場合、即座に AirManage を利用する為の再起動処理が行われます。

設定完了後に「保存」ボタンを押して下さい。また、再起動を行うことで初回アクセスを行います。

6.2 AirManage 登録

AirManage を利用する場合、以下が済んでいる必要があります。利用申請をしていない場合、インターネットに接続している OpenBlocks にて「AirManage」→「AirManage 登録」タブから下記の処理を行えるようになっています。

1. AirManage アカウントの作成
2. テナントの作成
3. OpenBlocks 本体の登録及びテナントへの所属

尚、弊社の WEB フォームから AirManage の利用申請が行われている場合には、全て済んでいる為実施する必要はありません。

6.2.1 AirManage アカウント作成

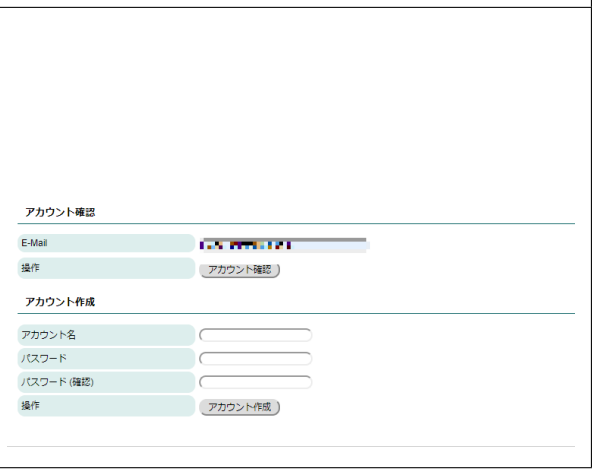
AirManage のアカウントを持っていない方のみアカウントを作成する必要があります。

● AirManage 登録 - アカウント確認	
項目	備考
E-Mail	AirManage サービスのログインアカウントに用いる E-Mail アドレスを入力します。
操作	「アカウント確認」ボタンを押すことにより、AirMange サービスのアカウントが存在しているか確認を行います。

AirManage のアカウントを持っていない方は以下のように表示されます。尚、既にアカウントを持っている方は表示されるメッセージに従い、”テナント作成 ”または ” OpenBlocks IoT Family 本体の登録及びテナントへの所属 ”を実施してください。

■アカウント未所持表示

■アカウント作成フォーム

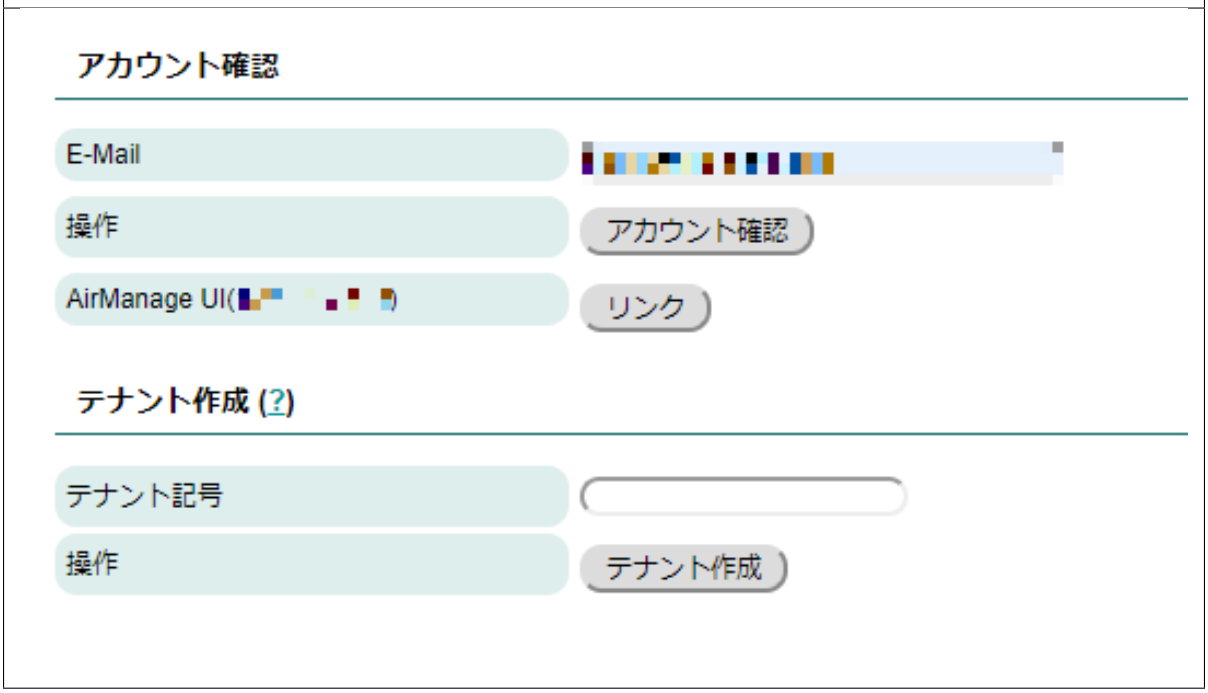


■アカウント作成完了 (ブラウザアクセス)

アカウント作成	
項目	備考
アカウント名	AirManage サービスにて表示されるアカウント名を入力します。
パスワード	AirManage サービスにログインする際のパスワードを入力します。尚、パスワードに使用可能な文字は ” ”(スペース) を除く半角英数字となります。また、文字通は最低 8 文字となります。
パスワード (確認)	AirManage サービスにログインする際の確認用のパスワードを入力します。
操作	「アカウント作成」ボタンを押すことにより、AirMange サービスのアカウント作成処理を行います。 作成処理が正常に行われた場合、設定している E-Mail アドレス宛に仮登録受付メールが届きます。メール本文内からのブラウザでアクセスすることで、本登録が行われます。

上記にて、AirManage のアカウントを作成した後からテナント作成を行っていない場合、アカウント確認にて以下のように表示されます。

■アカウント所持・テナント未作成表示



アカウント確認	
項目	備考
AirManage UI	「リンク」ボタンを押すことにより、使用する AirManage サービスのログイン WEB 画面が表示されます。今後アクセスに備え、ブックマークに登録することを推奨いたします。尚、項目名の () 内は使用する AirManage サービスの FQDN が表示されます。

6.2.2 テナント作成

AirManage のアカウントが完了しており、対象のアカウントがテナントをもっていない場合、「アカウント確認」を行った結果以下のように表示されます。

※既にテナントに所属しているアカウントの場合には以下のように表示されません。同一テナントにて複数のテナントに所属したい場合には、弊社サポート宛へご連絡下さい。

テナント作成	
項目	備考
テナント記号	ユーザーが所属するテナントをユニークに扱う為、記号 (名称) を入力します。 入力可能文字は英数字及び記号の"_"となります。
操作	「テナント作成」ボタンを押すことにより、テナントの作成処理を行われます。 既に同一名のテナントが存在する場合には作成できません。

また、テナントに所属している場合には以下のように表示されます。

テナント一覧	
項目	備考
テナント一覧	ユーザーが所属しているテナントがセレクトフォームに一覧で表示されます。

6.2.3 本体の登録及びテナントへの所属

AirManage のアカウントが完了及びテナント作成が完了している場合、「アカウント確認」を行った結果以下のように表示されます。

本体登録	
項目	備考
ノード名	AirManage のテナント内で OpenBlocks IoT Family 本体をユニークに識別する為のノード名を入力します。 入力可能文字は英数字及び記号の"_"となります。
操作	「本体登録」ボタンを押すことにより、OpenBlocks IoT Family 本体がテナント一覧で選択しているテナントに対して、入力したノード名で AirManage に登録されます。正常に登録できた場合、AirManage の初回アクセス設定部の設定を反映する確認のポップアップが表示されますので、「OK」等の了承するボタンを押してください。

登録が完了したら、"*AirManage* **初回アクセス設定**"を参考に AirManage の利用開始を行ってください。

注釈: 尚、過去に AirManage に登録したことがある本体にて、AirManage サービスの WEB 画面から対象のノードを削除したこと場合、本体登録が失敗します。その場合、再度対象の OpenBlocks にて AirManage を使用する場合には、弊社サポート宛にご連絡ください。

第7章　拡張機能

出荷直後状態の本筐体では、ネットワーク設定等を設定するソフトウェアのみ組み込まれています。IoT Gateway として使いたい場合等の機能拡張を行いたい場合には、『メンテナンス』→『機能拡張』から対応パッケージを追加することが行えます。

尚、WEB UI 本体のバージョンが古い場合には、機能拡張でインストールされるパッケージのバージョンの方が新しい可能性が発生することがあります。このような場合、正常に動作しない恐れがある為、機能拡張のパッケージインストール前に自動でアップデート後にインストール処理が行われます。

7.1 機能拡張用パッケージのインストール

WEB UI の「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

インストールしたいパッケージを選択し、インストールの「実行」ボタンを押すことでインストールされます。

※本機能にてインストールする場合、筐体がインターネット環境下である必要があります。

※インターネットに接続している回線が遅い場合には、パッケージのインストールに長時間かかることがあります。

実行ボタンを押した場合、確認ウインドウが表示されます。インストールするパッケージがあっている場合には、”OK ”等の確認を了承するボタンを押してください。

また、インストール中はボタン等が選択できなくなります。

※実行ボタンを押した後、状況確認ボタンが表示されます。このボタンを押すとインストール状況を確認できます。

インストール作業の成否問わず完了するとウィンドウが表示されます。

インストールに成功した場合には、ウィンドウメッセージを了承する旨のボタンを押してください。また、本機能にてインストール完了後、再起動が必要となりますので、本体再起動を行ってください。

※インストールに失敗した場合にはインターネット環境等を再確認し、再度インストールを実行してください。

※一部のパッケージのインストールには sources.list 及び Pub キーの追加が必用となる場合があります。

本機能からインストール可能なパッケージは、ドキュメント作成時現在以下となっています。

パッケージ	内容
Samba	Samba 用 WEB UI 及びファイル共有用ソフト一式となっています。本機能については「Samba 設定ガイド」を参照して下さい。
IoT データ制御	IoT データ制御用 WEB UI 及び各種アプリケーション一式となっています。本機能については「データハンドリングガイド」を参照して下さい。
Node-RED	Node-RED 用 WEB UI 及び Node-RED 一式となっています。本機能については「Node-RED スターターガイド」を参照して下さい。
セキュリティ	WEB UI や SSH への不正ログインに対してアクセス拒否塔を実施する機能一式となっています。本機能については「セキュリティガイド」を参照して下さい。
カメラ	カメラによる画像取込設定用 WEB UI 及び画像表示・動体検知ソフト一式となっています。本機能については「カメラ設定ガイド」を参照して下さい。
Docker	Docker DAEMON をインストールします。
Moby	Microsoft 社が構成した Docker DAEMON 一式をインストールします。
Docker(WEB UI 込み)	Docker コンテナ等を WEB UI から制御できる機能一式をインストールします。尚、Docker DAEMON についてもインストールします。本機能については「Docker 設定ガイド」を参照して下さい。
Azure IoT Edge	Azure IoT Edge 本体及び設定を行う WEB UI 一式をインストールします。Docker DAEMON が存在しない場合、Docker DAEMON についてもインストールします。本機能については「Azure IoT Edge 設定ガイド」を参照して下さい。
FTP	FTP サーバ DAEMON 及び設定用 WEB UI 一式をインストールします。本機能については「FTP 設定ガイド」を参照して下さい。

注釈: 本ドキュメント作成時点において、Moby 及び Docker パッケージの切り替えは自動では行えなません。そのため、ご使用したいパッケージに関しては WEB UI 込みでのインストール前に使用したい Docker のパッケージを意図的にインストールしてください。

第8章　設定項目別リファレンス

注釈: 本章にて実施している *root* **パスワードの設定** 項及び *WEB* **管理者パスワード変更** 項パスワード設定はセキュリティ上重要です。その為、クラックされにくくなるようなパスワードを設定してください。

8.1 サービス制御・拡張・AirManage 機能の表示/非表示

本 WEB-UI は IoT 関連向けにカスタマイズされていますが、別の目的に本装置を利用の際、サーバの基本設定部分のみ残し IoT サービス関連の WEB 表示を無効にすることが出来ます。

機能制御	
項目	備考
サービス機能	サービスタブを非表示にします。
拡張機能	拡張タブを非表示にします。
AirManage 機能	AirManage タブを非表示にします。

8.2 プロセス状況表示機能

ユーザーの追加したプロセスや基本的なプロセスの監視を行えます。

プロセス状況表示	
項目	備考
プロセス状況表示機能 (ユーザー定義)	例えば dhcpd 等の監視したいプロセスを登録しておくとダッシュボードにそのプロセスが起動しているか表示されます。最大 3 つまで登録できます。

8.3 ストレージアラート機能

定期的（1 時間に 1 回）にストレージ容量をチェックし、閾値を超えた場合にメールで通知させる機能です。ログ等によるストレージ容量の圧迫を監視できます

ストレージ管理 (メール通知)	
項目	備考
セルフチェック	本機能を使用する場合、「有効」を選択します。
閾値 (デフォルト 80%)	アラートを上げる際の閾値です。
SMTP サーバ：SMTP ポート	メールサーバのアドレスとポートを入力。SMTP Auth に対応したサーバを使用する場合、チェックを入れます。
SMTP Auth	「SMTP Auth を使う」にチェックを入れた場合に表示されます。SMTP Auth 用のユーザ名、パスワードを設定します。
送信元アドレス	メール送信の際の From アドレスを入力します。
宛先アドレス	メール送信の際の To アドレスを入力します。
テストメール	設定した内容でテストメールを送信します。メール本文の内容確認、設定に誤りがないかが確認できます。

8.4 root パスワードの設定

本装置に SSH やシリアルコンソールでログインする際に利用可能な root アカウントのパスワードを変更できます。

変更したいパスワードを確認欄と併せ 2 回入力し、「保存」ボタンを押します。

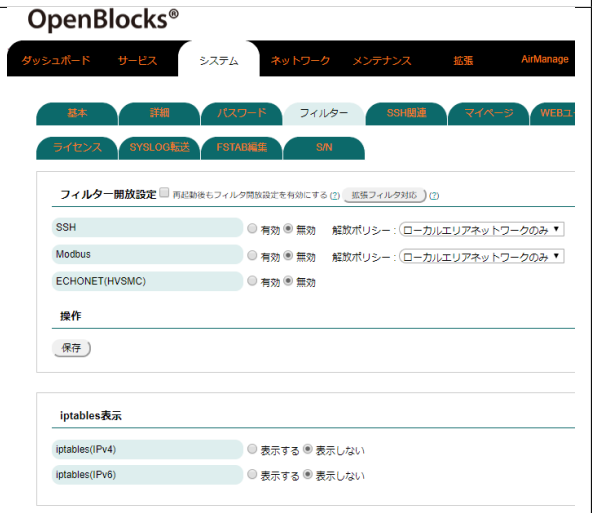
本システムを利用する際には、セキュリティ確保のために必ずデフォルトパスワードを変更してください。

注釈: 本装置のデフォルトの root アカウントのパスワードは 0BSI0T です。

(2 つある 0 は数字です。)

8.5 フィルター許可

本装置の各フィルターを一時的、または再起動後等の恒久的に有効にできます。

デフォルト	拡張機能をインストールした場合
	

項目	備考
フィルター開放設定	再起動後等も各フィルター開放を有効にする場合には、チェックを入れて保存ボタンを押します。
拡張フィルタ対応	ユーザーが独自に追加したアプリケーション等においてポート開放等が必要な場合、iptables 及び ip6tables コマンドにてポート開放を行う必要があります。ポート開放設定を行うシェルスクリプトファイルの雛形を作成します。
SSH	SSH を使って本装置にログインする時にラジオスイッチの有効を選択し保存ボタンを押します。
WEB UI(モバイル回線)	モバイル回線経由での WEB UI アクセスをする際に、ラジオスイッチの有効を選択し保存ボタンを押します。
【各種機能により追加される項目】	追加機能により項目が表示されます。SSH 等と同様にラジオボタンにて選択し保存ボタンを押してください。

解放ポリシーは、"ローカルネットワークのみ"または"全て"から選択してください。モバイル回線モデムを搭載しているモデルの場合は、“ モバイル回線のみ ”が選択可能となります。

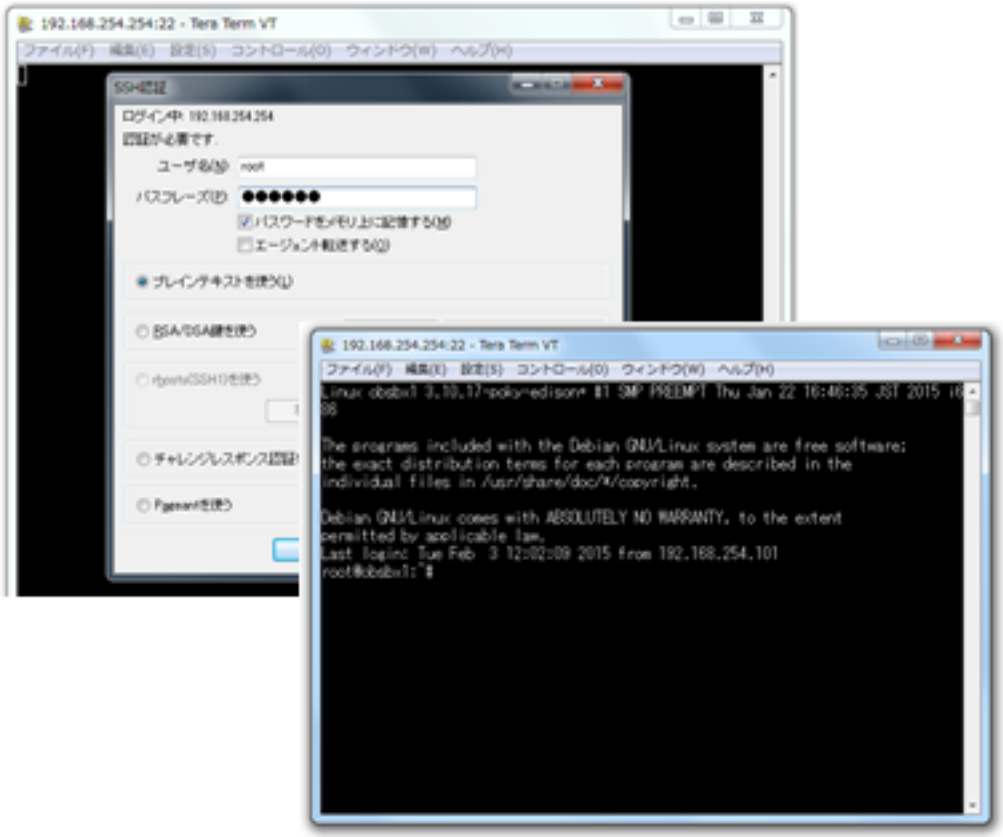
また、一部のサービスについては解放ポリシーが存在しない物があります。

※モバイル回線モデムが搭載されている場合に表示されます。通常では WEB UI へのアクセスは WLAN または Ethernet 経由でのアクセスのみサポートしています。モバイル回線経由のアクセスはセキュリティ上、通常サポートしていません。

項目	備考
拡張フィルタ設定ファイル編集	拡張フィルタに対応させた場合、上図のようポート開放設定を行うシェルスクリプトファイルの入力フォームが表示されます。対象ポートへのアクセスを許容する iptables 及び ip6tables コマンドを入力後、保存ボタンを押すことで反映されます。 ※本機能では iptables 及び ip6tables コマンドの理解が必用となりますので、ご注意ください。

iptables 表示	
項目	備考
iptables(IPv4)	ラジオボタンを表示するに設定すると iptables の IPv4 の内容を表示します。
iptables(IPv6)	ラジオボタンを表示するに設定すると iptables の IPv6 の内容を表示します。

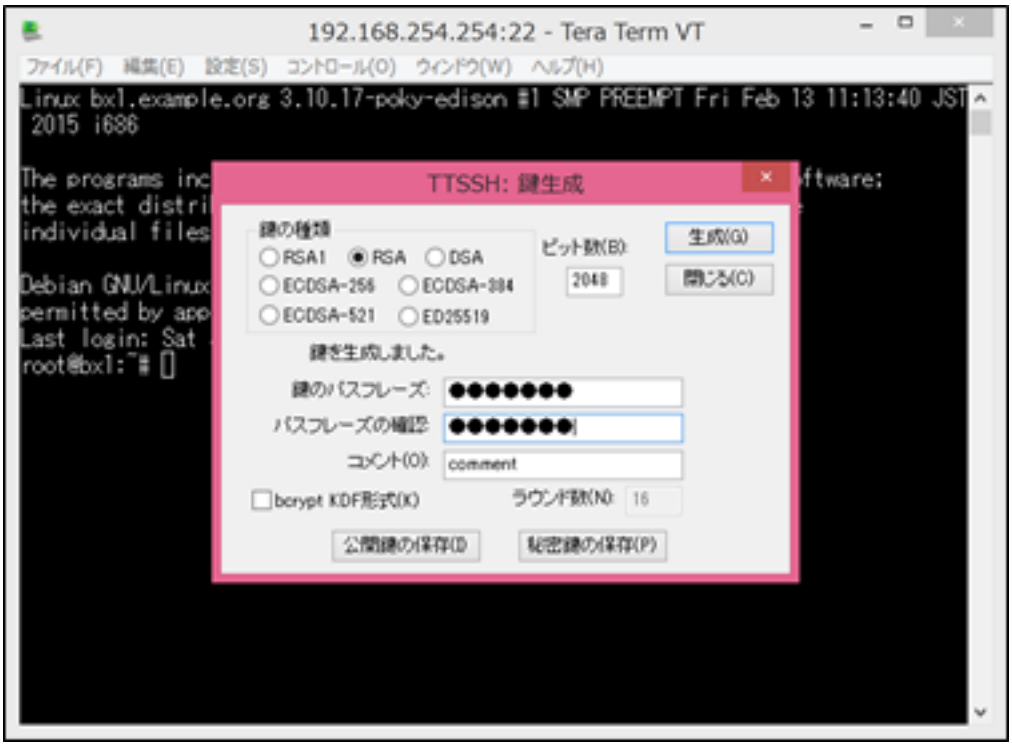
注釈: 各フィルター開放が不要になった場合、無効化を忘れないでください !!



SSH は上図の通り、TeraTerm などのターミナルソフトで IP アドレスを指定してログインします。また、SSH をよりセキュアに運用するためには「 *SSH* **の鍵交換** 」で解説される公開鍵の登録を行うことをお奨めします。

8.6 SSH の鍵交換

SSH をよりセキュアに使う為の設定画面です。



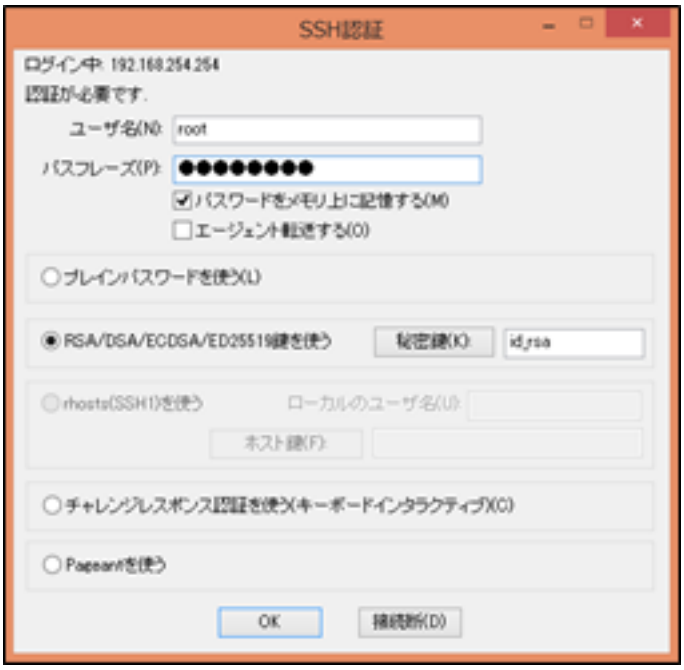
先ず、上画面のように TeraTerm などで公開鍵・秘密鍵を生成します。

TeraTerm の場合、指定ディレクトリにこの 2 つの鍵が保存されるので、そのうち公開鍵をテキストエディタなどで表示し、コピーバッファに保存してください。

設定箇所はシステム⇒ SSH 関連タブとなります。

SSH 設定	
項目	備考
SSH ポート番号	SSH に使用するポート番号を設定します。
root ログイン許可設定	本装置に root アカウントでの SSH ログインを許可する場合に「パスワード認証不可」を選択します。
パスワード認証	SSH に鍵を使わずアクセスする場合は、パスワード認証を「許可」します。鍵を使った認証にする場合には、「禁止」を設定します。
公開鍵	前述の TeraTerm などで作った公開鍵を貼り付けてください。なお、鍵を使わない時には空欄にしておきます。

設定が完了したら「保存」ボタンを押します。



以上の設定後、SSH での鍵付きのログインを行ってください。

上画面は TeraTerm での接続例です。

8.7 WEB 管理者パスワード変更

WEB UI の管理者パスワードが変更できます。尚、ユーザ名の変更はできません。

設定箇所はシステム⇒マイページタブとなります。

編集後、保存ボタンを押した時点で変更が有効になります。

変更後はログインし直してください。

8.8 WEB ユーザー

WEB UI のログインユーザーの追加や、別のログインユーザーのパスワード変更 (スーパーユーザーのみ) が行えます。

設定箇所はシステム⇒ WEB ユーザータブとなります。

ユーザー名、パスワード等を設定後、保存ボタンを押した時点で変更が有効になります。

8.9 ファイル管理

WEB UI を用いて OpenBlocks 内の特定ディレクトリにファイルのアップロード等が行えます。

設定箇所はシステム⇒ファイル管理タブとなります。

ダウンロード、削除、移動、実行権付与または編集をする場合には、ファイルを選択し、ボタンを実行内容のボタンを押してください。

また、アップロードする場合には、「ファイルを選択」からアップロードするファイルを選択後に「アップロード」ボタンを押してください。

尚、アップロード先は以下となります。

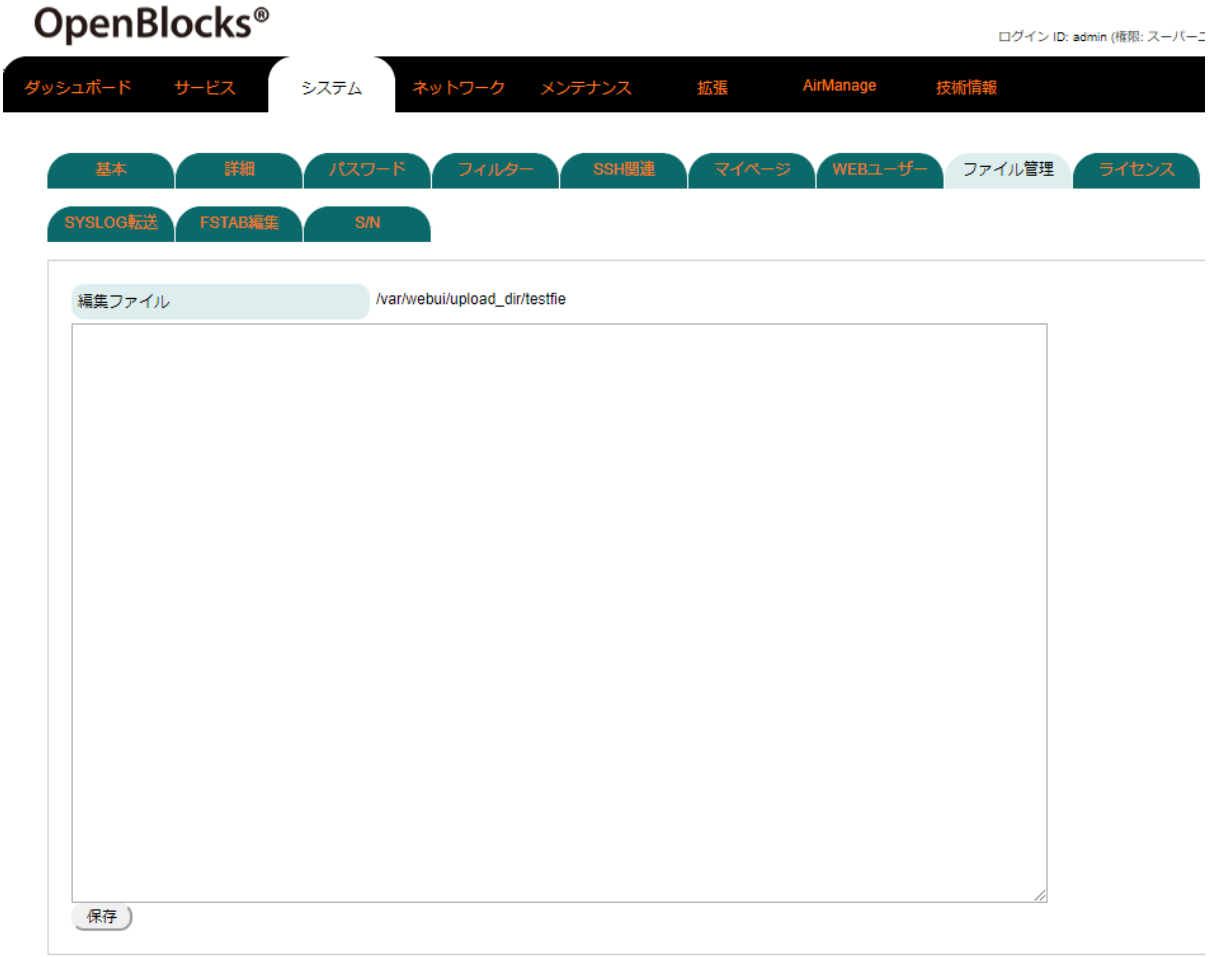
Dir : /var/webui/upload_dir/

容量が 256MB を超えるファイルはアップロードが行えません。そのようなファイルをアップロードする場合には SSH を有効にし、SFTP にてファイルをアップロードしてください。

新規ファイル及び新規ディレクトリ生成は、ファイルまたはディレクトリパスを入力し作成します。また、/var/webui/upload_dir/下にファイル作成が可能です。(上位のディレクトリ下には作成できません。)

一括エクスポートは/var/webui/upload_dir/下の各ファイル一式を tar+gz 形式に圧縮したファイルがエクスポートされます。

一括インポートは/var/webui/upload_dir/下に tar+gz 形式のデータを展開します。



ファイル選択後、編集ボタンを押した場合には左図のように画面が表示されます。

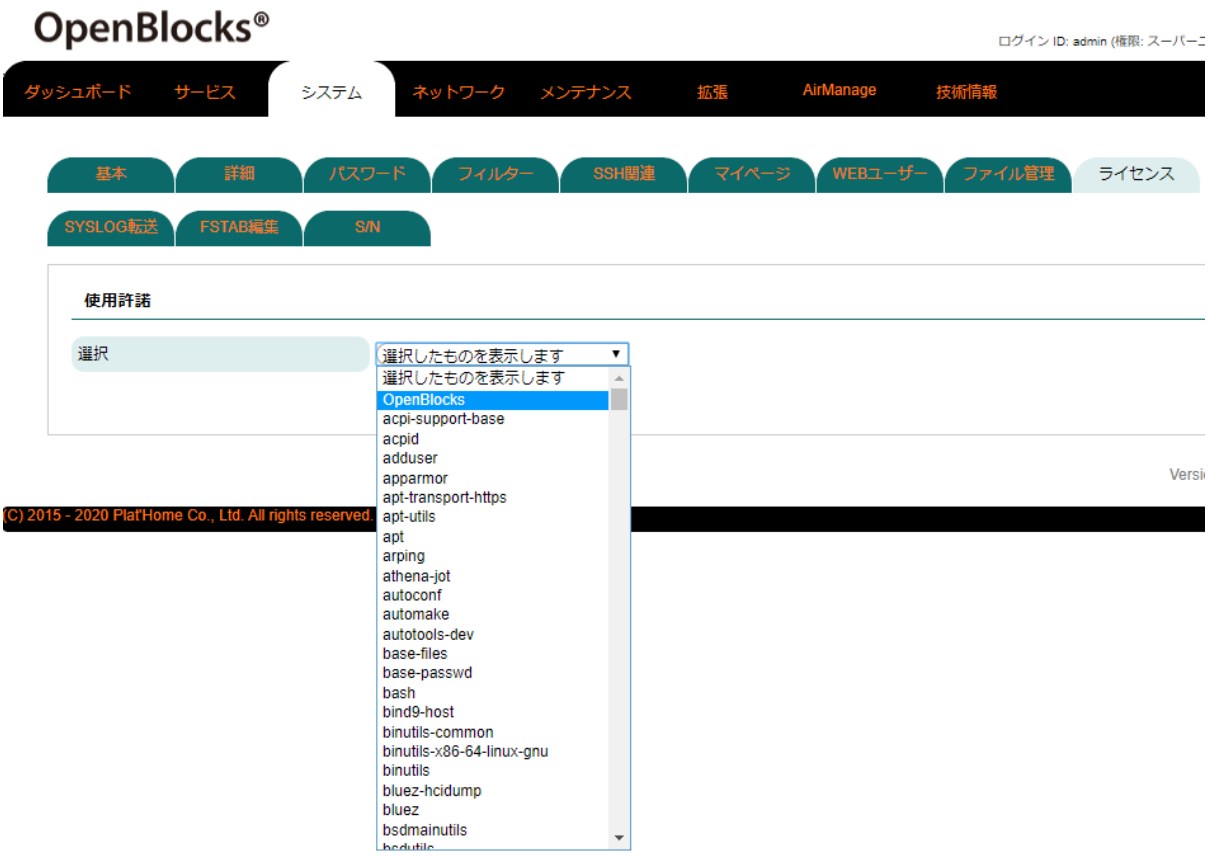
編集内容を保存する場合には、編集ボタンを押してください。

尚、編集はテキストファイルのみサポートします。

8.10 ソフトウェアライセンスの表示

WEB UI にて使用されているソフトウェアライセンス、使用許諾を表示できます。

表示箇所はシステム⇒ライセンスタブとなります。



本装置に使用されているソフトウェアライセンス、使用許諾をソフトウェア毎にプルダウンメニューから選んで表示できます。

8.11 本体シリアルの確認

WEB UI にて OpenBlocks 本体のシリアル番号を確認できます。

確認箇所はシステム⇒ S/N タブとなります。

※上図で表示されているシリアルはサンプルです。

8.12 ダイナミック DNS

WEB UI にてダイナミック DNS サーバに対して、現状の IP アドレスを定期的に登録します。

設定箇所はネットワーク⇒ダイナミック DNS タブとなります。

ダイナミック DNS	
項目	備考
使用設定	ダイナミック DNS を使う時に「使用する」を選択します。
DDNS サービス	DDNS サービスを選択します。
ユーザー名	DDNS のユーザアカウントを入力します。
パスワード	DDNS のパスワードを入力します。
FQDN	DDNS 上に登録された FQDN を入力します。
登録 IP 情報	DDNS 上に通知する IP アドレスの属性を設定します。

設定が完了したら「保存」ボタンを押します。設定内容を反映させるには装置の再起動が必要です。

8.13 静的ルーティングの追加

AP モード時などのルータ動作時に静的ルーティングの設定が必要な時ここで設定します。

設定箇所はネットワーク⇒ルーティングタブとなります。

ネットワークアドレスとネットマスクを指定し、ゲートウェイとなる装置の IP アドレスを指定し保存ボタンを押します。

静的ルーティングは複数登録が出来ます。

設定内容を反映させるには装置の再起動が必要です。

8.14 通信確認

ネットワークが使えているか ping コマンドなどでテストできます。

テスト箇所はネットワーク⇒疎通確認タブとなります。

使用するコマンドはプルダウンメニューで ping / traceroute / nslookup から選択できます。

コマンドを選択し実行ボタンを押すと下部に実行結果が即表示されます。

8.15 ネットワーク状態確認

ネットワークの様々な状態を確認できます。

確認箇所はネットワーク⇒状態タブとなります。

本装置の設定を一通り終わり、再起動した後にこの画面で確認する事をお奨めします。

また、以下の項目を確認できます。

・IP アドレス

・無線 I/F 情報 (無線 I/F 対応モデルのみ)

・ルーティング情報

・arp 情報

・ホスト情報

・DNS サーバ情報

・モデム情報 (モデム搭載モデルのみ)

・SIM 情報 (モデム搭載モデルのみ)

8.16 コンフィグレーションのバックアップとリストア

WEB UI にて設定したコンフィグレーションを WEB クライアントに対してバックアップを行えます。また、そのファイルを用いてリストアが実施できます。

実行箇所はメンテナンス⇒設定タブとなります。

エクスポートの実行ボタンを押すと、コンフィグレーションファイルのバックアップを WEB クライアントにダウンロードします。

設定をリストアする時には、インポートのファイル選択で、バックアップファイルを選び、実行ボタンを押すとコンフィグレーションファイルをもとにリストアされます。

※本装置のシステムセットアップが完了した際、設定を変更した際は都度バックアップの実行を推奨します。

※コンフィグレーションファイルの編集は原則サポートいたしません。

※コンフィグレーションファイルのインポートにおいて、以下の置換ルールが適用されます。

置換元文字列	置換内容	備考
@@SERIAL@@	本体シリアル番号	

8.17 システムソフトウェアのアップデート

本装置のファームウェアや OS、アプリケーションのバージョンアップを確認し、アップデートできます。

実行箇所はメンテナンス⇒システム更新タブとなります。

本装置がインターネット接続環境にある場合はオンラインアップデートが可能です。

オンラインにある「更新有無を確認」を押すとリポジトリ情報に基づいてアップデート内容を確認し、更新があれば本画面の下部にそれぞれのアップデート内容が表示されるので、更新する場合はアップデートを実行してください。

尚、オフラインパッケージはインパクトあるアップデート時に弊社から提供するパッケージです。

WEB クライアント (ファイルサイズ上、PC を推奨) にダウンロードして、オフラインにある「ファイルを選択」ボタンで PC 上にあるアップデートパッケージを選んで実行ボタンを押します。

セキュリティのアップデートは頻繁にあるので、なるべくマメにアップデートを行うことを推奨します。

また、適用パッケージによっては再起動後にアップデートが反映されるものが多数ありますので、アップデート後は本体再起動の実施を強く推奨いたします。

8.18 SMS 送信

本装置は一部のモバイル回線モデムモジュールにて SMS をサポートしています。

(モバイル回線契約に SMS 機能が無い場合、サポートできません。また、本装置に SIM が挿入されている必要があります。)

これにより、SMS を WEB UI 上から送信することが可能となっております。

SMS 送信	
項目	備考
宛先電話番号	SMS 送信先の電話番号を入力します。
本文	送信する SMS の本文を入力します。尚、本文には最大 70 文字まで入力可能です。

電話番号及び本文を入力し、「送信する」ボタンを押すことにより SMS が送信されます。

8.19 SSH トンネル

SSH サーバに対して SSH 接続を行い、トンネルを構築します。これにより、SSH サーバからトンネル経由にて OpenBlocks 側へ SSH アクセスを行うことが可能となります。

※本機能を使用する場合には、「 **フィルター許可** 」にて SSH のフィルターを許可しておく必要があります。

OpenBlocks®
ログイン ID: a
ダッシュボード
サービス
システム
ネットワーク
メンテナンス
拡張
AirManage
技術情報
スクリプト編集
コマンド実行
SMSコマンド実行
SMS送信
SSHトンネル
監視
URIプロキシ
稼働情報
SSHトンネル (?)
使用設定
使用する
使用しない
SSHトンネルモード
常時接続
SMSコントロールイベント
ログインユーザー
SSH接続先ホスト
SSH接続先ポート
SSH折返用ポート
SSH認証設定
パスワード認証
鍵認証
パスワード
操作
保存

SSH トンネル	
項目	備考
使用設定	本機能を使用するか設定します。使用する場合には「使用する」を選択してください。
SSH トンネルモード	SSH トンネルを構築するモードを設定します。 "常時接続"にした場合、稼働中は常に SSH トンネルの構築を試みます。 "SMS コントロールイベント"に設定した場合、SMS または SMS コントロールダイレクト実行により SSH トンネルが構築されます。 ※ SMS の場合、最長 30 分間 SSH トンネルが構築されます。
ログインユーザー	SSH サーバにてログインするユーザーを指定します。
SSH 接続先ホスト	接続先の SSH サーバの IP アドレスや FQDN を設定します。
SSH 接続先ポート	接続先の SSH サーバのポート番号を設定します。通常は 22 番となります。
SSH 折返用ポート	SSH サーバにて接続元の本機器へアクセスする為のポート番号を設定します。
SSH 認証設定	SSH サーバへ接続する際の認証方式を設定します。
パスワード	認証方式がパスワード認証の場合のパスワードを入力します。
パスフレーズ	認証方式が鍵認証の場合、パスフレーズを入力します。
プライベートキーファイル	認証方式が鍵認証の場合、プライベートキーファイルパスを入力します。

※鍵認証におけるプライベートキーファイルはファイル管理からアップロードしてください。

設定完了後、保存ボタンを押してください。また、再起動することにより本機能は有効となります。

8.20 サポート情報

サービスに関するサポート窓口情報に関して、メンテナンス⇒サポートタブにて確認が行えます。

※上図はサンプル画像となります。

連絡先等の変更の恐れがあります。最新の情報は WEB UI にて確認を行ってください。

ログ・環境情報取得のダウンロードの実行ボタンにてサポートに必要なログ情報等が取得できます。

ログ情報等の取得は WEB UI 等で標準的に弊社でのサポート時に必要な情報のみが含まれております。

この部分に独自アプリケーションのログ等を含めたい場合、「システム」→「ファイル管理」にて"add_support.list"というファイルをアップロードし、ファイル内に追加したいデータパスを記載することですることでサポート用データに追加の情報の拡張が行えます。

※ add_support.list サンプル
/var/log/apt /usr/src/

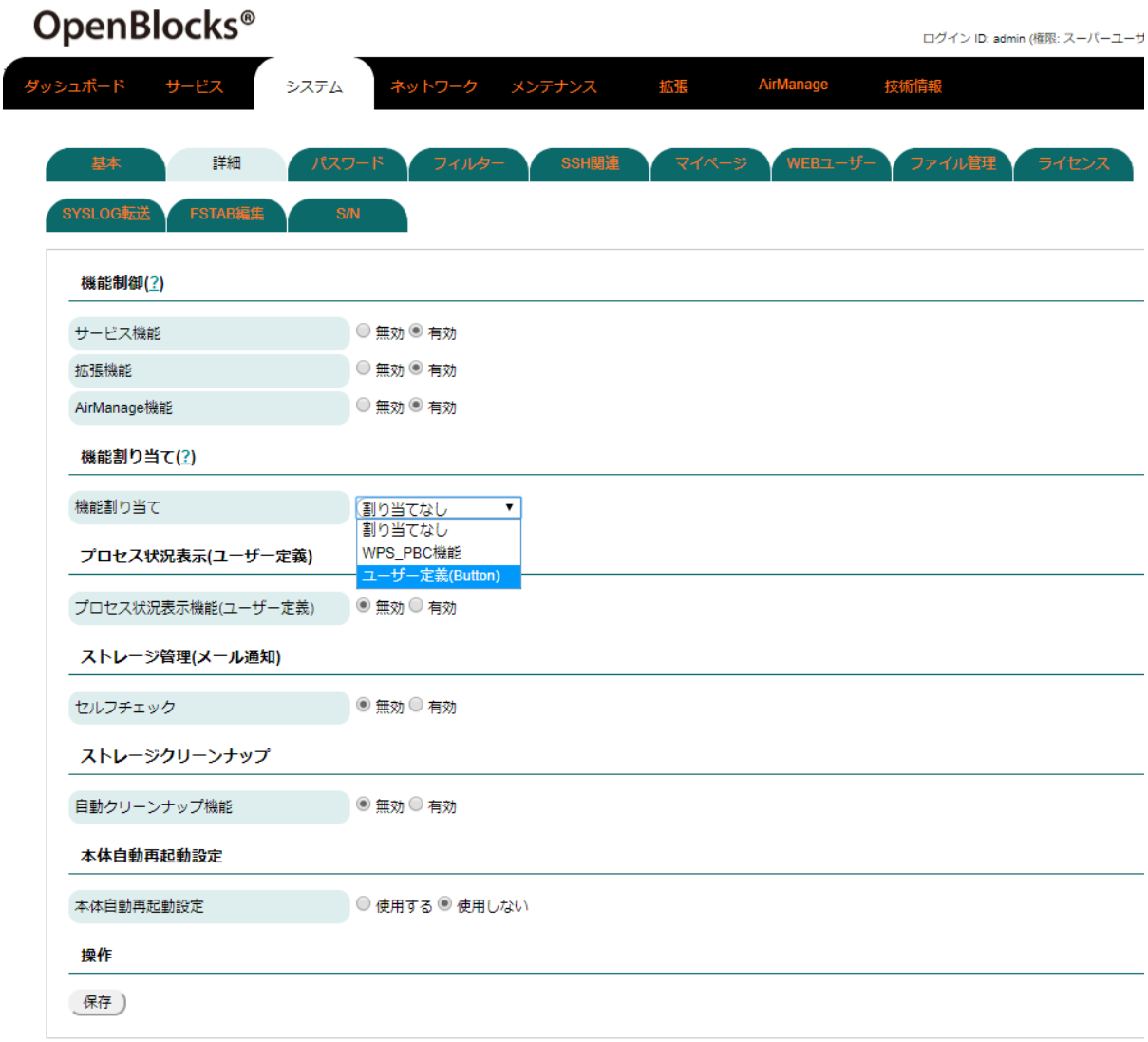
尚、ルートディレクトリ (/) 及び 1 階層目 (/tmp や/var 等) の指定は無視されます。

8.21 OpenBlocks のサービス及び技術情報一覧

「技術情報」タブをクリックすると当社 HP への各リンクが表示されます。(アクセスする場合にはインターネット接続が必用となります。)

各リンクは AirManage サービスへの加入ページや、製品マニュアル及び OpenBlocks を用いた技術情報サイト、FW アップデート情報、購入後製品に関するお問い合わせへとなります。

8.22 FUNC スイッチの機能割当



「システム」の「詳細」タブにて FUNC スイッチに対して機能が割り当てできます。以下の機能を設定可能です。

・割り当てなし

・WPS_PBC 機能

・ユーザー定義 (Button)

WPS_PBC 機能は WLAN インターフェースが存在している場合のみ、使用可能です。

WLAN0 インターフェースを使用状態にした場合、WPS_PBC 機能が有効となります。

※ AP モードの場合には、WPS_PBC の設定を行う必要があります。

※クライアントモードの場合において、WPS_PBC 機能にて AP とのネゴシエーション完了後、設定反映の為自動で再起動が行われます。尚、モデム使用時や他の I/F が DHCP 設定の場合には使用不可となります。

また、ユーザー定義 (Button) は拡張機能のスクリプト編集にて、対象のスクリプトを作成している場合に機能が有効となります。

8.23 監視機能

OpenBlocks 内のログファイルや稼働プロセスの監視を行えます。

ログファイル監視は特定キーワードが出力された場合にアテンション喚起状態となります。

また、プロセス監視は設定したプロセスが稼働していない場合にアテンション喚起状態となります。尚、対象プロセスの未稼働状態でアテンション喚起状態となったプロセスは監視対象外となります。

アテンション喚起状態のリセットはダッシュボードから実施可能です。

また、本機能は AirManage 機能と連動しており、AirManage 機能を有効にしている場合には AirManage リモート管理サーバ側にてアテンション情報の確認することが出来ます。

監視機能を有効にする場合、使用設定を「使用する」を選択します。

ログ監視機能を有効にする場合には、ログ監視部の使用設定を「使用する」を選択します。

また、プロセス監視を有効にする場合には、プロセス監視部の使用設定を「使用する」を選択します。

ログ監視	
項目	備考
使用設定	ログ監視機能を有効にする場合には、ログ監視部の使用設定を「使用する」を選択します。使用しない場合は「使用しない」を選択します。
ログ監視設定	追加ボタンにて監視設定の項目を追加することが出来ます。(最大 8 個までとなります)
ログファイルパス	監視対象とするログのファイルパスを設定します。 (ex. /var/log/messages)
アラート対象文字列	アラート (アテンション) として扱う文字列を設定します。複数の条件を設定する場合、"|" にて区切ることで設定可能となります。 (ex. error|ERROR)

プロセス監視	
項目	備考
使用設定	プロセス監視機能を有効にする場合には、プロセス監視部の使用設定を「使用する」を選択します。使用しない場合は「使用しない」を選択します。
プロセス監視設定	追加ボタンにて監視設定の項目を追加することが出来ます。(最大 8 個までとなります)
監視プロセス	監視対象とするプロセスを設定します。 正確にチェックする場合には、パスを含んだ状態で設定することを推奨します。

設定完了後、「保存」ボタンを押すことで監視設定が完了となります。

「保存」ボタンを押した場合には既にアテンション喚起状態となっていた場合には、解除されます。

また、アテンション状態はダッシュボードで確認が行えます。

●アテンション未発生の場合

●アテンション喚起状態の場合

エラー解除ボタンを押すことにより、アテンション喚起状態を解除できます。

また、アテンション喚起のログが一定行以上となった場合、全件表示ボタンが表示されます。そのボタンを押すことにより、設定している監視状態のログが確認できます。

8.24 URI プロキシ機能

OpenBlocks 内の WEB プロセスエンジン機能により、WEB UI 経由にて自ホストまたは他ホストの WEB へアクセス可能です。

本機能を設定することにより、WEB UI のポートのみで自ホストや別ホストの WEB プロセスへアクセスできますので、セキュリティ観点上本機能の使用を推奨します。

また本体内で稼働している及び稼働可能な WEB プロセス等については **注意事項及び補足** の 使用ポート一覧 を参照して下さい。

「拡張」の「URI プロキシ」タブにて WEB UI 経由にてアクセスしたい WEB サービスを設定します。

項目	備考
Prot.	アクセス先の WEB ページのプロトコルを"http"または"https"から選択します。
URI	ユニークな URI を設定してください。 ※英数字のみサポートしております。 Ex.) Node-RED の例 : nodered
IP	アクセスしたい WEB サービスが稼働している自ホストまたは他ホストを IPv4 形式の IP アドレスで指定してください。
Port	アクセスしたい WEB サービスが稼働している PORT 番号を設定してください。 Ex.) Node-RED のデフォルト例 : 1880
EXT_URI	アクセスしたい WEB への追加 URI を設定することが可能です。特定の URI にアクセスした場合には設定してください。 Ex.) Node-RED の/worldmap 設定例 : worldmap

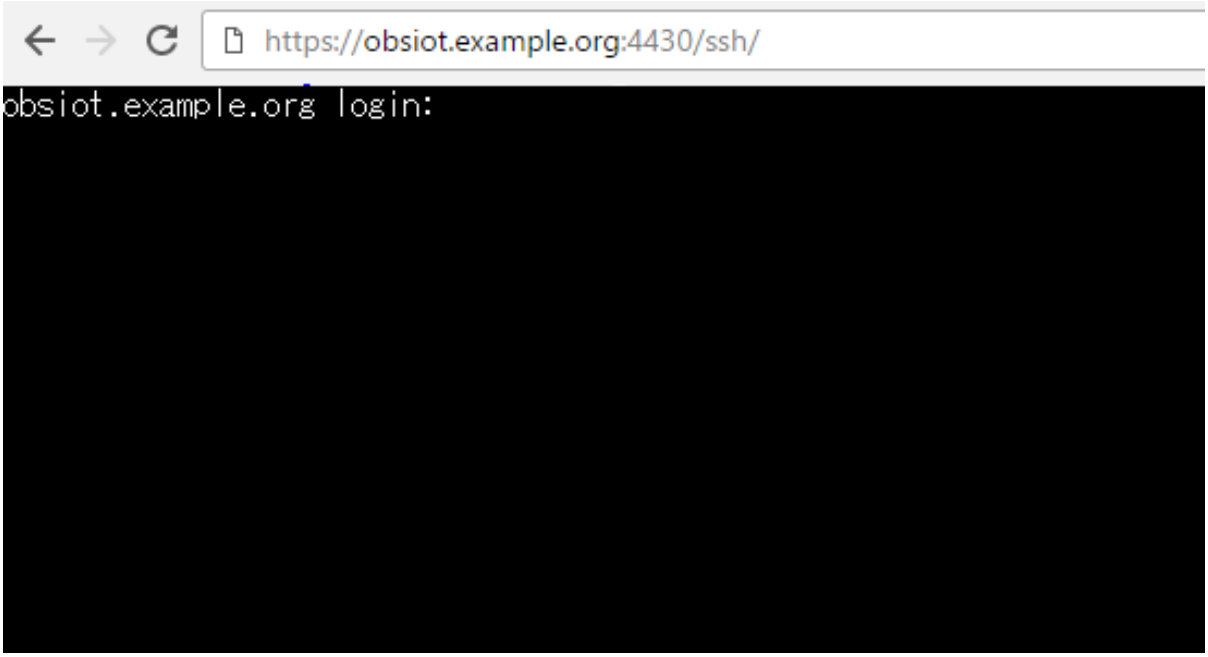
また、本機能で参照する WEB サービスのプロトコルが異なる場合、アクセス可否が異なります。以下の表をご確認ください。

WEB UI アクセスプロトコル	参照 WEB サービスプロトコル	アクセス可否
HTTP	HTTP	可能
HTTP	HTTPS	不可能
HTTPS	HTTP	可能
HTTPS	HTTPS	可能

8.25 WEB コンソール機能

OpenBlocks IoT Family 内に shell in a box が起動しています。このプロセスは WEB ブラウザ経由でコンソール機能が使用可能となります。本機能では 4200 番ポートを使用しております。セキュリティの関係上、本ポートをデフォルトで開放する機能は用意しておりません。そのため、URI プロキシ機能を用いてアクセスを行ってください。

※ HTTPS 経由でのアクセスのみ対応しています。



上図は対象の画面となります。

本機能向けに sudo 機能をすべて有効にしているアカウントを用意しております。尚、root アカウントでのログインは行えません。

アカウント : obsroot

パスワード : 0BSI0T ※ 0 は数字の 0 です。

※パスワードはクラックされる恐れがありますので、passwd コマンドにて変更してください。

8.26 SYSLOG 転送機能

本製品内にて出力される全ての SYSLOG を外部の SYSLOG サーバへ転送が行えます。

「システム」→「SYSLOG 転送」タブから設定が行えます。

SYSLOG 転送	
項目	備考
転送機能	SYSLOG 転送の機能設定を行います。転送を行う場合には、「使用する」を選択してください。
転送プロトコル	SYSLOG 転送を行う際のプロトコルを「TCP」または「UDP」から選択します。
転送ホスト	SYSLOG 転送先のホストを IP アドレスまたは FQDN 形式で設定します。
転送ポート	SYSLOG 転送先のポート番号を設定します。通常は 514 から変更する必要はありません。

設定完了後に「保存」ボタンを押すことで、反映されます。

8.27 ストレージクリーンナップ機能

本機能はストレージが閾値を超えた場合において、特定ディレクトリ配下の優先保存期間を超えている過去ファイルを削除します。

本項は「システム」→「詳細」タブから設定してください。

ストレージクリーンナップ	
項目	備考
自動クリーンナップ機能	本機能を使用する場合は「有効」を選択してください。使用しない場合は「無効」を選択してください。
対象ディレクトリ	クリーンナップ対象とするファイルを格納するディレクトリを設定してください。 重要なファイル (コマンドやライブラリ) が存在するディレクトリは指定しないでください。
閾値	本機能を適用する際の閾値とするストレージの使用率を指定します。
優先保存期間	何日以内のファイルを残すかを指定します。ここで指定した日数以内のファイル、削除されません。

8.28 電源監視機能

OpenBlocks IoT VX2 及び EX1 では内蔵バッテリーモジュールを用いることが可能です。このモジュールを用いている場合、停電等によって AC 等の電源供給が一時的にダウンした状態での動作が可能となります。電源供給がダウンした状態で一定時間経過後、シャットダウン処理 (電源喪失後実行コマンド) が行われます。

本項は「システム」→「詳細」タブから設定してください。

※本機能の表示は内蔵バッテリー対応しているモデルのみ表示されます。そのため、バッテリー搭載の有無とは連動しておりません。

※本機能は電源供給を確認し、内蔵バッテリー稼働後に安定したシステムのシャットダウンを行う機能です。内蔵バッテリーモジュールを搭載している場合、本機能を使用せずとも一時的な動作は可能となりますが、安定したシステムシャットダウンは行われません。

電源監視	
項目	備考
電源監視使用設定	本機能を使用するかを設定します。
電源喪失後コマンド発呼時間 [sec]	内蔵バッテリー稼働運用状態に切り替わった後の電源喪失後実行コマンドを実行する時間を設定します。
電源喪失後実行コマンド	安定したシステムのシャットダウンを行うコマンド (/sbin/poweroff) を設定してください。また、システムのシャットダウンのみではなく他の処理も同時に行う場合には、スクリプトを作成し指定してください。

8.29 本体自動再起動機能

連続稼働により自作アプリ等がメモリーリークを起こし、OS 全体が不安定となることがあります。この場合、定期的に本体の再起動を定期的に行うことによって安定稼働を行うことができます。

本項は「システム」→「詳細」タブから設定してください。

毎日指定	曜日指定	日にち指定

本体自動再起動設定	
項目	備考
本体自動再起動設定	OpenBlocks 本体を定期的に再起動を行うかを指定します。本機能を使用する場合には「使用する」を設定してください。
再起動トリガー	"毎日", "曜日指定", "日にち指定"から選択可能です。
曜日指定 (曜日指定時に表示)	再起動を実施する曜日を選択します。
日にち指定 (日にち指定時に表示)	再起動を実施する日にちを選択します。
再起動実施時間	本体の再起動を行う時間を指定します。

8.30 Pub キー追加機能

アップデート等により参照する標準以外のリポジトリを使用するケースがあります。この場合、OpenBlocks 本体に対象のリポジトリのパブリックキーを追加する必要があります。また、意図せずパブリックキーを削除した場合にも本機能を用いて追加を行ってください。

本項は「メンテナンス」→「Pub キー追加」タブから設定してください。

※追加可能な Public キーが存在しない場合、対象タブは表示されません。

※ Public キーのみではなく、参照するリポジトリを追加する必要があります。この場合、AirManage 等によって source.list の変更を推奨いたします。

Pub キー追加	
項目	備考
Pub キー追加	追加するパブリックキーを選択してください。
説明	対象のパブリックキーの説明が表示されます。
インストール	選択されているパブリックキーのインストール作業を実施します。本機能ではインターネット通信を行うため、事前にネットワークの設定をインターネット通信を行えるようにしてください。

8.31 HTTP プロキシ(クライアント)機能

本機能は HTTP プロキシでのみインターネットに出るネットワークでの OpenBlocks を運用する際に用いる機能となります。

本項は「ネットワーク」→「HTTP プロキシ」タブから設定してください。

※本機能はモバイル回線等に切り替わった場合でも設定は有効となります。この場合、設定に合わないネットワーク環境となる為、通信が行えなくなります。そのため、モバイル回線との併用はしないでください。

HTTP プロキシ	
項目	備考
使用設定	HTTP プロキシ経由で通信を行うかを設定します。HTTP プロキシ経由で通信を行う場合に「使用する」を設定してください。
プロキシサーバー	経由するプロキシサーバーを指定します。
プロキシ用ポート	経由するプロキシサーバーへアクセスする際のポートを指定します。
プロキシユーザー	プロキシサーバーへアクセスする際の Basic 認証が必用な場合、ユーザー名を指定します。
パスワード	プロキシサーバーへアクセスする際の Basic 認証が必用な場合、ユーザーに対応するパスワードを指定します。
非プロキシ経由アクセスホスト	プロキシ経由でアクセスを行わないホストを複数設定できます。複数指定する場合は、","区切りにて指定してください。 ※ネットワークでの指定は行えません。

8.32 停止・再起動

本装置の停止または再起動を行う事ができます。

実行箇所はメンテナンス⇒停止・再起動タブとなります。

	停止実行ボタン遷移後	再起動実行ボタン遷移後

停止・再起動	
項目	備考
停止	本装置を停止することができます。停止の実行ボタンを押すことにより、停止画面に遷移します。停止画面の実行ボタンを押すことにより、本装置の停止処理が行われます。 また、停止処理を実行した場合、自動では復旧いたしません。そのため、実運用時は行わないでください。
再起動	本装置を再起動することができます。再起動の実行ボタンを押すことにより、再起動画面に遷移します。再起動画面の実行ボタンを押すことにより、本装置の再起動処理が行われます。

8.33 フォーマット機能

OpenBlocks IX9 の SSD 未搭載モデルにて、追加で SSD を搭載した場合にフォーマット機能が使用可能となります。実行箇所はシステム⇒フォーマットタブとなります。

本機能は追加の SSD のパーティションに対してフォーマット処理が行えます。本機能を使用する前にパーティションの作成を推奨いたします。

フォーマット	
項目	備考
パーティション	追加搭載した SSD のパーティションを選択します。※パーティションが存在しない場合、ブロックデバイスが選択できます。
ファイルシステム	ext4/ext3/ext2/ntfs/vfat から選択してください。

パーティション及びファイルシステム選択後、実行ボタンを押すことによりフォーマット処理が行われます。

8.34 簡易マウント機能

OpenBlocks IX9 の SSD 未搭載モデルにて、追加で SSD を搭載した場合に簡易マウント機能が使用可能となります。実行箇所はシステム⇒簡易マウントタブとなります。

本機能は追加の SSD のパーティションのマウント設定を簡易的に行えます。

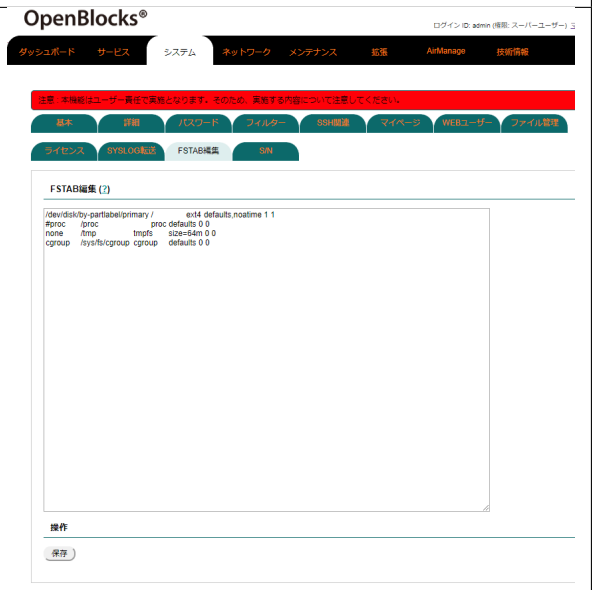
簡易マウント	
項目	備考
パーティション	マウント対象のパーティションを選択します。 マウント可能なファイルシステムは ext4/ext3/ext2 となりますので、フォーマット機能にて対象のファイルシステムへとフォーマットしてください。
マウント先	対象パーティションのマウントポイントを指定します。

実行ボタンを押すことにより設定が保存されます。次回起動時に反映されます。

8.35 FSTAB 編集機能

OS が参照するファイルシステムの情報設定を編集することができます。本機能により追加ストレージのファイルシステム参照を変更することが行えますが、設定が間違っている場合正常に起動できない場合が発生します。そのため、本機能を用いて変更を行う場合にはご注意ください。

実行箇所はシステム⇒ FSTAB 編集タブとなります。

編集確認前	編集確認後
	

FSTAB 編集	
項目	備考
パーティション	「確認」ボタンを押すことにより、本機能を用いるかの確認メッセージのポップアップが表示されます。 ポップアップにて ”OK ”を選択した場合、fstab ファイル内容がテキストエリアに表示されます。 このテキストエリアを編集し、保存することで編集が完了します。 尚、設定した fstab ファイル内容については次回起動時から反映されますので、設定変更後は再起動を行ってください。

8.36 ホスト追加機能

ローカルネットとワーク等の DNS を用いずホスト名と IP アドレスの解決を行う hosts ファイル内容について追記が行えます。

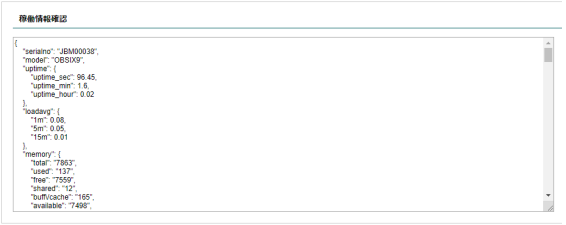
実行箇所はネットワーク⇒ホスト追加タブとなります。

ホスト追加	
項目	備考
ホスト追加	IP アドレスとホスト名を 1 行ずつ記載します。IP アドレスとホスト名の区切りはスペースにてを用いてください。 尚、ホスト名は複数記載することが可能です。 【書式】 <IP アドレス> <ホスト名> Ex.) 172.16.7.1 gateway

8.37 稼働情報機能

OpenBlocks 本体の稼働情報を定期的に取得し保存することができます。

実行箇所は拡張⇒稼働情報タブとなります。

<table>
<tr><td></td><td>稼働情報ファイルが取得できている場合</td></tr>
<tr><td>
</td><td> 
</td></tr>
<tr><td></td><td>※表示内容は稼働情報ファイル内の本体に依存する情報ののみとなります。</td></tr>
</table>

稼働情報	
項目	備考
使用設定	OpenBlocks 本体の稼働情報を取得する場合、「使用する」を選択し保存してください。
情報取得間隔	稼働情報の取得間隔を選択します。1 分毎/5 分毎/10 分毎/30 分毎/1 時間毎/2 時間毎/3 時間毎/4 時間毎/6 時間毎/12 時間毎から選択可能です。
一括操作	各取得項目について ”一括有効 ”または ” 一括無効 ”ボタンを押すことによって、対象の操作が行えます。
取得項目 [シリアル No.]	稼働情報に本体のシリアル No. を含めるかを"無効"、"有効"から選択します。
取得項目 [HW モデル]	稼働情報に本体のモデル情報を含めるかを"無効"、"有効"から選択します。
取得項目 [稼働時間]	稼働情報に本体の稼働時間情報を含めるかを"無効"、"有効"から選択します。
取得項目 [平均負荷]	稼働情報に本体の平均負荷情報を含めるかを"無効"、"有効"から選択します。
取得項目 [メモリ情報]	稼働情報に本体のメモリ情報を含めるかを"無効"、"有効"から選択します。
取得項目 [ストレージ情報]	稼働情報に本体の情報を含めるかを"無効"、"有効"から選択します。
取得項目 [NIC]	稼働情報に本体の NIC 情報を含めるかを"無効"、"有効"から選択します。
取得項目 [IPv4 ルーティング]	稼働情報に本体の IPv4 ルーティング情報を含めるかを"無効"、"有効"から選択します。
取得項目 [IPv6 ルーティング情報]	稼働情報に本体の IPv6 ルーティング情報を含めるかを"無効"、"有効"から選択します。
取得項目 [シリアル No.]	稼働情報に本体のシリアル No. を含めるかを"無効"、"有効"から選択します。
(取得項目 [WLAN 情報])	稼働情報に本体の WLAN 情報を含めるかを"無効"、"有効"から選択します。尚、本項目は本体に WLAN I/F 搭載モデルにのみ表示されます。
(取得項目 [BT 情報])	稼働情報に本体の BT 情報を含めるかを"無効"、"有効"から選択します。尚、本項目は本体に BT I/F 搭載モデルにのみ表示されます。
(取得項目 [モバイル情報])	稼働情報に本体のモバイル情報を含めるかを"無効"、"有効"から選択します。尚、本項目は本体にモデム搭載モデルにのみ表示されます。
固定情報付与	稼働情報に固定で付与したい情報を JSON で記載します。 Ex.) { "debug" : "test" }

稼働情報は以下に JSON ファイルとして保存しています。ファイル : /tmp/device_status.json

第9章　注意事項及び補足

9.1 OpenBlocks IoT VX シリーズの電源について

本製品は AC アダプタによる給電及びワイドレンジ電源入力以外での、電源運用は保障対象外となります。そのため、使用電源についてご注意ください。

9.2 自動再起動機能

本 WEB-UI はモバイル回線のモデムを制御しています。モバイル回線のモデムが不慮の復旧不能状態に陥った場合、本体再起動が動作します。

9.3 LTE/3G モジュール (ソフトバンク) 運用時のアクセス

LTE/3G モジュール (ソフトバンク) を運用している場合において、LTE 回線側にグローバル IP アドレスが付与される場合には、グローバル IP アドレスと以下のポート番号の関係から各種サービスが使用できます。

※フィルター許可にて開放している必要がありますのでご注意ください。この場合、再起動後も適用している必要があります。

※グローバル IP アドレスは DDNS サービスを用いることで容易に使用できます。

サービス種類	ポート番号	補足
SSH	50022	
WEB UI(HTTP アクセス)	50880	ブラウザでのアクセスとなります。
WEB UI(HTTPS アクセス)	54430	ブラウザでのアクセスとなります。

9.4 GRUB メニュー表示方法について

OpenBlocks IX9 及び OpenBlocks IoT VX2 では、起動時にシリアルポートに GRUB メニューが表示されます。そのため、GRUB メニューの確認等を行いたい場合は、下図のようにシリアルコンソール接続にて作業を行ってください。

Windows PC の場合、USB ポートに接続されると自動的に USB シリアルドライバがインストールされます。(Windows PC がインターネット環境につながっている場合です。)

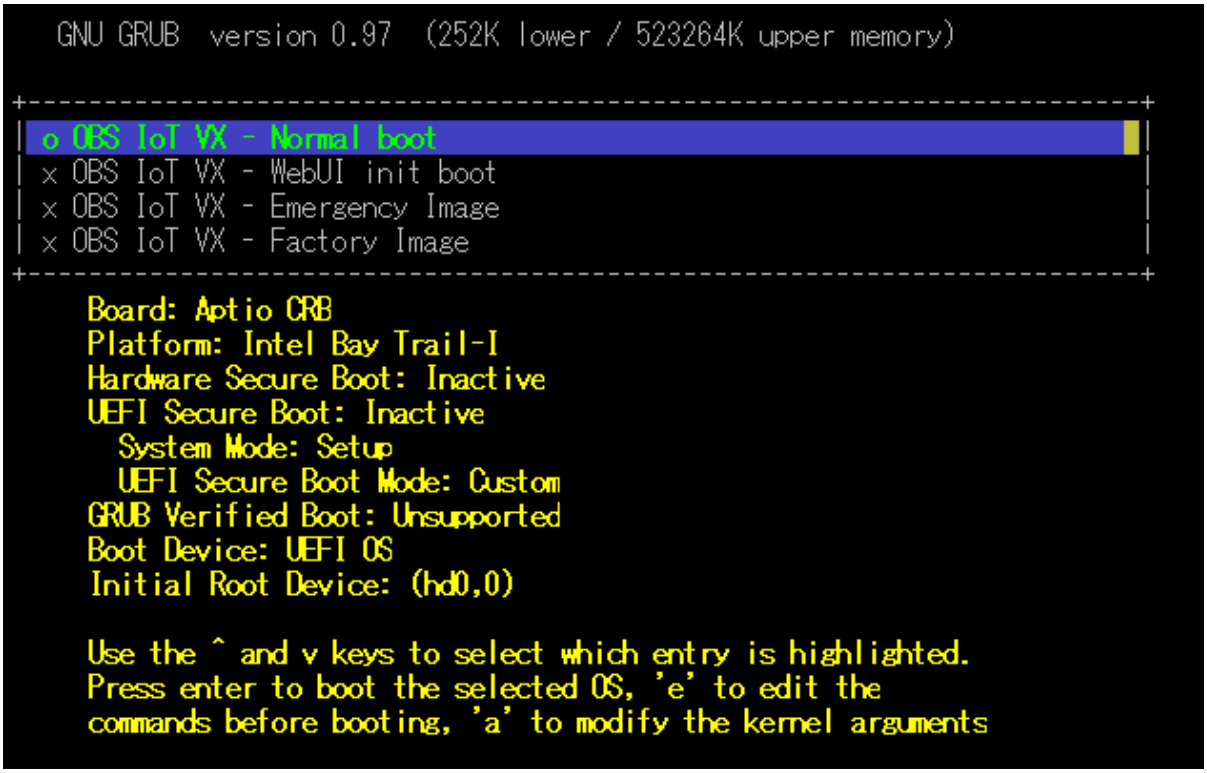
ドライバのインストールが完了したら、TeraTerm や PuTTY などのターミナルソフトでシリアルポート接続が可能となります。

尚、OpenBlocks のシリアルポートのデフォルト通信パラメータは以下の通りです。 通信速度:115200bps

データ長：8bit

パリティ：無し

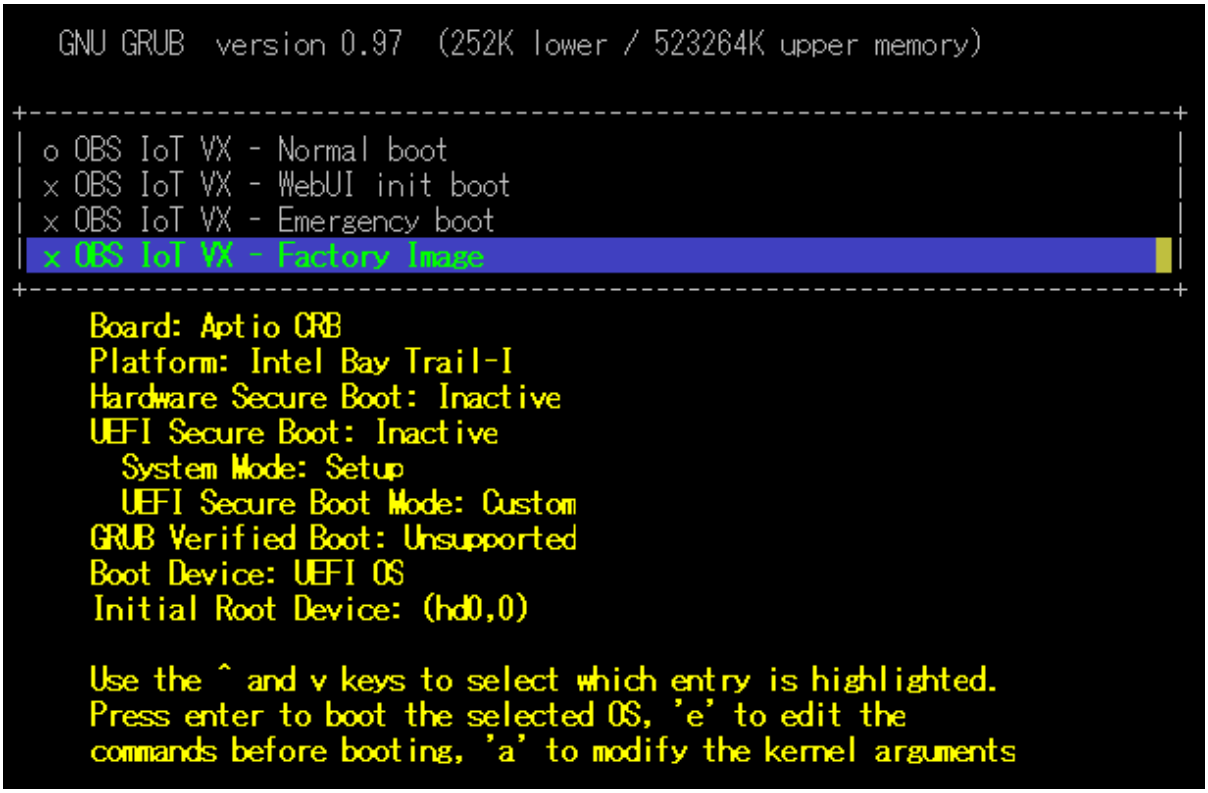
ストップ：1bit



9.5 Factory Reset(工場出荷状態への切り替え)

OpenBlocks IX9 及び OpenBlocks IoT VX2 にてストレージ領域へパッケージの追加や重要データの削除等を実施してしまい、工場出荷状態に戻したい場合、GRUB メニューの「Factory Image」を選択することで工場出荷状態へ戻すことが出来ます。

工場出荷状態に戻した場合には、設定したデータ等は削除されますのでご注意ください。



また、OpenBlocks IoT BX/EX シリーズにて工場出荷状態に戻したい場合には弊社製品 HP の『ドキュメント』→『その他』→『ファクトリーリセット』をご確認し、作業を実施してください。

9.6 使用ポート一覧

WEB UI 込みでの OpenBlocks では以下のポートを使用及び使用する可能性があります。

サービス種類	ポート番号	補足
FTP	21	FTP インストール時
SSH	22	ポート番号変更可能
DNS	53	
DHCP	67	
NTP	123	
NetBIOS	137	Samba インストール時 (UDP)
NetBIOS	138	Samba インストール時 (UDP)
NetBIOS	139	Samba インストール時
Samba	445	Samba インストール時
Modbus	502	IoT データ制御インストール時
WEB UI(HTTP アクセス)	880	
ECHONET	3610	IoT データ制御/HVSMC 使用時
Node-RED	1880	Node-RED インストール時 (ポート番号変更可能)
Shell in a box(WEB SSH)	4200	
WEB UI(HTTPS アクセス)	4430	
SSH	50022	LTE/3G モジュール (ソフトバンク) / WAN 側のみ
WEB UI(HTTP アクセス)	50880	LTE/3G モジュール (ソフトバンク) / WAN 側のみ
WEB UI(HTTPS アクセス)	54430	LTE/3G モジュール (ソフトバンク) / WAN 側のみ
WEB UI 独自サービス	63003	

9.7 自動外部ストレージマウント機能

WEB UI において特定のボリュームラベルの付いたデバイスが見つかった場合、自動でマウントされます。

WEB UI の機能等で保存先管理等を行う場合にご使用ください。

ボリュームラベル	マウント先	備考
WEBUI_STORAGE	/var/tmp/storage	ファイルシステムは NTFS のみをご利用いただけます。

第IV部

開発者向けガイド

第10章　はじめに

本書は、OpenBlocks シリーズの開発者向けガイドです。

利用者向けについては、OpenBlocks シリーズ向け WEB UI セットアップガイドを参照してください。

第11章　ご利用の前に

11.1 製品概要

OpenBlocks シリーズは OS として Debian GNU/Linux を採用した汎用サーバー製品です。ハードウェア特性を活かす為のカスタマイズを行っていますが、本カスタマイズ以外は、Debian やその他 Linux の一般的な操作方法でご利用いただけます。

11.2 SSD ベースのシステム開発における注意点

近年、フラッシュメモリーのコストダウンによってスマートフォンを始めノート PC や様々な機器では、従来のハードディスクに代わって SSD が採用されています。また、本製品に使用されている eMMC は SSD の一種です。

SSD はランダムアクセス性能が高速であり、かつ、ソリッドステートである利点からメカニカルな障害耐性や環境性能が非常に良くなりますが、その反面、ハードディスクに比べてデータの書き換え制限が大幅に少なくなります。

SSD には大きく分けると SLC と MLC にわかれ、書き換え性能が数万回の SLC タイプが数ギガの容量帯では主流でしたが、マルチビットセルで書き換え回数数千回の低コストの MLC が大容量化を実現し、スマートフォンや PC に多く使われるようになり、現在では SLC の SSD は徐々に姿を消し始めています。

当社マイクロサーバー製品のオプションでも SLC 製品は一部だけ小容量帯で残すのみです。

ゆえに MLC の SSD 搭載でのマイクロサーバー運用が大変多くなります。

MLC では概ね 3000 回の書き換え性能があり、3000 回を超えた辺りからビットエラーが発生し始めますが ECC エラー回復します。

しかし、その ECC 回復可能な条件を超えるとリードエラーを発生します。

ゆえにこの状況に追い込まないようなシステム作りが必要になります。

●セルあたりの書き換え数とフラッシュメモリーのブロックサイズ

セルあたり 3000 回と言っても、SSD に対する書き込みが 1 バイトでも 1 回は 1 回となります。

最近のフラッシュメモリーは少ないアドレス線で大容量をサポートするため 512KB くらいの大きなブロックで読み書きされます。

つまり、書き込みにおいて 1 バイト書いても 512KB 書いても書き換えは 1 回なのです。

ゆえに SSD への書き込みは、小さなデータサイズで細かくよりも、なるべくバッファに多くデータを貯めこんで大きなデータサイズで一気に書く方が SSD に対する書き込み回数を最小限にすることが出来ます。

●ウェアレベリング機能

SSD は書き換え制限が少ないため、同じブロックアドレスを何回も書き換える動作に対して、同一の実ブロックアドレスへの書き換えを集中させないように平均化しています。

これはブロックアドレスを仮想化して実現しています。

OS は SSD に対して使っているブロックと使われていないブロックを通知しており SSD は次に書き込みがあった場合、一番書き換え数の少ないブロックをその書き込みのために準備しています。

これによって結果的に書き換え数が平均化されます。

●スタティックウェアレベリング

従来のウェアレベリングの場合、使われている領域でかつほとんど書き換えのないデータが例えば 50%あった場合、OS のインストール時から 1 回だけ書かれたブロックが 50%を占め、その部分がほぼ新品のまま、再利用されている残りの 50%の領域がどんどん書き換えられてしまい、全体で平均化した場合よりも 2 倍早く SSD の寿命が来てしまいます。

この対策に考えられたのがスタティックウェアレベリングで、ほとんど書き換えられていないブロックのデータを再利用の多いブロックに移動し、ほぼ新品のままだったブロックを再利用領域にあてがいます。

これにより、50%ほぼ書き換えのないブロックがあっても、ほぼ SSD の寿命をまっとうできます。

● SSD の全体の書き換え回数を想定する

例えば 4GB の SSD で 512KB ブロックタイプであった場合、セルあたり 3000 回の書き換えできるとして 512KB 以下のデータを書いた場合、SSD の総書き換え回数を想定すると以下の様になります。

4294967296B ÷ 524288 = 8192 (4GB ÷ 512KB）

物理ブロック数は 8192 個あることになります。

それぞれが 3000 回書き換えられるとしたら

8192 × 3000 = 24576000 回

つまり 1 回ずつ 1 バイトのデータを書くとたったの 18.4MB 書くと寿命がつきてしまいます。（実際にはこういった書き込みは SSD 側で効率化してます）さらに 512KB サイズの書き込みを想定すると、ブロックのセグメントがキッチリ 512KB の境界をまたがないでならば 1 回の書き込みですが、OS からのファイルアクセスでは、ブロックの途中から書かれるケースもあるので、512KB 以下のデータ書きとはいえ、半分の確率で 2 回書き込まれます。

つまり

24576000 回 × 75 % = 18432000 回 (これは 512KB だけの書き込みを想定)

さらに OS からのアクセスではもう一つ書き換え回数が追加されます。

それはファイルのクローズ処理によるファイルコントロールブロックの更新で最低 1 回書き換えが発生します。

もちろん SSD ではキャッシュを使ったりしてさらに書き換え回数を低減しておりますが、基本的にはこういった処理系になります。

●なるべく大きな SSD を使用する

例えば前記の 4GB の SSD では 8192 個のブロックが存在しますが、8GB ならその 2 倍の 16384 個のブロックが存在するので書き換え回数が単純に 2 倍になります。

SSD が同じブロックサイズならサイズに応じて単純に比例して書き換え回数がアップするわけです。

ゆえに可能な限りサイズの大きい SSD を利用すれば書き換え回数によるトラブルへの耐性が上がっていくわけです。

● tmpfs の利用で SSD 書き込みを軽減する

Linux システムの場合、何の考慮もしないでシステム開発すると、ストレージは無限に使えるデバイスとしての前提で利用されてしまいます。

データ保存の必要もなくともストレージ領域をワーキングのためのバッファとして普通に使われてしまいます。

こういった事で SSD の寿命を縮める事の無いように、ワーキング処理に必要なストレージはなるべく tmpfs に配置する様、考慮してシステムを設計して行きます。

また、オープンソースのソフトウェアでも、自分用に確保したストレージ領域をテンポラリーとして利用するケースも多く、この場合は、そのファイルを tmpfs の領域にリンクを張って対処します。

●ログ

Linux システムでは何事においてもストレージにログを残す事が一般的ですが非常に細かくログを残すプロセスがある場合が、一度 tmpfs に書かせてから定期的にまとめてログを SSD に移すなどの工夫が必要です。

こういった対策では、突然の電源断には対応出来ませんが、この点はトレードオフしてあきらめるか、例えば UPS 対策されたシスログサーバーにログを送信するかそう言った考慮を検討する必要があります。

11.3 SIM について

OpenBlocks シリーズにて、搭載可能な SIM 形状は一部の型番の物を除き mini-SIM(2FF) です。mini-SIM 対応モデルにて micro-SIM 及び nano-SIM を使用する場合には、脱落防止フィルム有及び接着テープ有で SIM を固定できるアダプタを使用してください。尚、SIM アダプタを使用した場合での SIM スロットの破損は有償修理対象となります為、ご注意ください。

11.4 eMMC ストレージパーティション情報

本装置に使用されている eMMC のパーティション情報は以下となります。

● OpenBlocks IoT VX2 の場合

番号	format	サイズ	OBS 用途	デバイス名
1	fat16	1.5GB	Boot	mmcblk0p1
2	ext4	30.5GB	Primary	mmcblk0p2

● OpenBlocks IX9(eMMC ブート) の場合

番号	format	サイズ	OBS 用途	デバイス名
1	fat16	1.5GB	Boot	mmcblk0p1
2	ext4	6.2GB	Primary	mmcblk0p2

● OpenBlocks IX9(SSD ブート) の場合

番号	format	サイズ	OBS 用途	デバイス名
1	fat16	1.5GB	Boot	sda1
2	ext4	120GB	Primary	sda2

※ sda2 のサイズについては、搭載ストレージ量に依存します。

● OpenBlocks IoT BX 及び EX シリーズの場合

番号	format	サイズ	OBS 用途	デバイス名
1		1MB	通常起動用 u-boot	mmcblk0p1
2		2MB	通常起動用 u-boot 環境変数	mmcblk0p2
3		1MB	非常時起動用 u-boot	mmcblk0p3
4		2MB	非常時起動用 u-boot 環境変数	mmcblk0p4
5	ext2	1MB	シリアル番号、MAC アドレス等	mmcblk0p5
6		1MB	カーネルパニックメッセージ保存用	mmcblk0p6
7	fat16	210MB	通常起動用 FW 用	mmcblk0p7
8	ext4	105MB	バックアップ領域	mmcblk0p8
9	fat16	210MB	リカバリー起動用 FW 用	mmcblk0p9
10	ext4	3351MB	Primary	mmcblk0p10

11.5 ストレージモード

基本的なユーザーランドデータを eMMC または SSD から参照し動作を行います。不意な電源断が起こった場合、物理ストレージ上のファイルが破損する恐れがありますが、Docker 等のアプリケーション側では unionfs にてストレージ上のデータ参照を行いますので問題なく稼働が行えます。

唐突な電源断によるファイル破損は、主に書き込み中のファイルが対象となります。そのため、基本システムへの影響は発生しにくいよう通常書き込むファイルはログファイル程度に抑えるのを推奨いたします。

mount コマンドの実行結果
root@obsix9:~# mount /dev/mmcblk0p2 on / type ext4 (rw,noatime) devtmpfs on /dev type devtmpfs (rw,relatime,size=4015372k,nr_inodes=1003843,mode=755) sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime) proc on /proc type proc (rw,nosuid,nodev,noexec,relatime) securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime) tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev) devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000) tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755) tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k) tmpfs on /sys/fs/cgroup type tmpfs (rw,mode=755) cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate) cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd) cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids) cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio) cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices) cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory) cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio) cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct) cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event) cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset) cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb) cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer) hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M) mqueue on /dev/mqueue type mqueue (rw,relatime) debugfs on /sys/kernel/debug type debugfs (rw,relatime) systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=33,pgrp=1,timeout=0,minproto=5,maxproto=5,direct) none on /tmp type tmpfs (rw,relatime,size=65536k) nginx_dir on /var/log/nginx type tmpfs (rw,relatime,size=16384k) monitlog on /var/webui/monitlog type tmpfs (rw,relatime,size=5120k) tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=805180k,mode=700)

11.6 PC と OpenBlocks の接続

OpenBlocks と PC を付属の USB ケーブルを用いて接続します。

※ OpenBlocks IoT VX2 ではワイドレンジ電源入力についてもサポートしておりますので、そちらでの稼働についてもご検討下さい。

Windows PC の場合、USB ポートに接続されると自動的に USB シリアルドライバがインストールされます。(Windows PC がインターネット環境につながっている場合です。) ドライバのインストールが完了したら、TeraTerm や PuTTY などのターミナルソフトでシリアルポート接続が可能となります。

尚、OpenBlocks のシリアルポートのデフォルト通信パラメータは以下の通りです。

通信速度：115200bps

データ長：8bit

パリティ：無し

ストップ：1bit

通信が確立後、起動が完了するとログインプロンプトが表示されます。デフォルトの root 権限でログイン操作を開始して下さい。

login：root

password：0BSI0T

※パスワードはデフォルトの場合となります。WEB UI にてパスワードを変更している場合には、設定したパスワードを用いてください。

11.7 WEB UI

本システムには基本的なシステム設定が行えるように WEB UI を搭載しています。

ネットワーク設定やモバイル回線の制御等は、WEB UI から実施してください。

また、WEB UI から各種機能の追加等が行えますので、基本的な使用方法等については WEB UI セットアップガイドを参照してください。追加した機能については各種機能のマニュアルを参照してください。

第12章　ご利用方法

本装置は Debian GNU/Linux を採用した汎用サーバー製品です。そのため、通常の Debian と同様にご利用いただいて構いません。また、本項では本装置特有の情報を記載しております。

12.1 ステータスインジケーターの表示色

OpenBlocks シリーズの正面にある LED は RGB の組み合せで 7 色に点灯し、それぞれの表示色で点滅などをスクリプト制御しています。

WEB UI が使用されている場合には、デフォルトの LED 点灯仕様は以下の通りになっています。

ステータスインジケーター			
状態	色	点灯状態	備考
本体及び OS 起動中	黄	点灯	本体起動及び OS 起動が終わるとモバイル回線の電波受信チェックへ移行します。※ SIM が挿入されていない場合は緑点滅。
SIM スロット未使用	緑	点滅	SIM が無い状態での正常稼働または電波受信待機状態への移行待ち状態。
モバイル回線電波：強	白	点滅	「開発者ガイド」の電波状態詳細参照。
モバイル回線電波：中	水色	点滅	「開発者ガイド」の電波状態詳細参照。
モバイル回線電波：弱	青	点滅	「開発者ガイド」の電波状態詳細参照。この電波強度での通信はリトライが多発する可能性があります。そのため、モバイル回線を使用する場合にはなるべく電波強度が中以上の状態にて使用してください。
モバイル回線電波：圏外	紫	点滅	「開発者ガイド」の電波状態詳細参照。
FUNC ボタンによる機能有効時	黄	点滅	モバイル回線や SIM スロット未使用時におけるステータスインジケーターと交互点滅となります。
OS 終了中	黄	点灯	
AirManage 初回アクセス失敗時	赤	点灯	AirManage リモート管理サーバへの初回アクセスが失敗した際に表示となります。WEB UI 未使用時の場合は 5 分後に OS が終了します。
強制 SIM モード時におけるモデムデバイスファイルまたは SIM カード認識失敗時	赤	点灯	モデム搭載モデルにおいて強制 SIM モードを有効にし、起動時にモデムのデバイスファイルが存在しないまたは SIM カードが認識できない場合の表示となります。5 分後に OS が再起動します。
AirManage 及び SIM 認識失敗時の OS 終了中	赤	点灯	

電波状態詳細				
HW / モデム種別	電波：強	電波：中	電波：弱	電波：圏外
BX1	-87dBm 以上	-88〜-108dBm	-109〜-112dBm	-113dBm 以下
BX3	-87dBm 以上	-88〜-108dBm	-109〜-112dBm	-113dBm 以下
BX5	-95dbm 以上	-95.1〜-105dBm	-105.1〜-120dBm	-120.1dBm 以下
LTE モジュール (KDDI)	アンテナ 2 本以上	アンテナ 1 本時	アンテナ 0 本時	圏外時
LTE モジュール (NTT ドコモ)	アンテナ 3 本時	アンテナ 2 本時	アンテナ 1 本時	圏外時
LTE/3G モジュール (ソフトバンク)	AT+CSQ 値: 14 以上	AT+CSQ 値: 13 〜 3	AT+CSQ 値: 2 〜 1	AT+CSQ 値: 0 または 99
LTE/3G モジュール (NTT ドコモ/KDDI)	-95dbm 以上	-95.1〜-105dBm	-105.1〜-120dBm	-120.1dBm 以下
BWA モジュール	-95dbm 以上	-95.1〜-105dBm	-105.1〜-120dBm	-120.1dBm 以下
3G モジュール (NTT ドコモ)	-87dBm 以上	-88〜-108dBm	-109〜-112dBm	-113dBm 以下
3G モジュール (ソフトバンク)	-87dBm 以上	-88〜-108dBm	-109〜-112dBm	-113dBm 以下

電波状態判定	
HW / モデム種別	電波状態判定
BX1	RSSI による判定
BX3	RSSI による判定
BX5	RSRP による判定
LTE モジュール (KDDI)	モデムモジュールによる判定
LTE モジュール (NTT ドコモ)	モデムモジュールによる判定
LTE/3G モジュール (ソフトバンク)	AT+CSQ 値判定
LTE/3G モジュール (NTT ドコモ/KDDI)	RSRP による判定
BWA モジュール	RSRP による判定
3G モジュール (NTT ドコモ)	RSSI による判定
3G モジュール (ソフトバンク)	RSSI による判定

WEB UI 使用時の LED 点灯制御スクリプト

/var/webui/bin/set_signal_value.sh

/var/webui/scripts/led_updater.sh

● LED の制御

LED の表示色・点灯状態を変更する場合は、/tmp/.runled ファイルの内容を編集します。

尚、WEB UI 使用時における SIM 挿入時は電波強度と連動する為、定期的に本ファイルが更新されます。

そのため、意図的に変更する場合は、WEB UI を未使用とするか LED 点灯制御スクリプトを終了させてください。

行	設定内容	備考
1 行目	点灯 1 時間 (msec)	1 以上
2 行目	点灯 2 時間 (msec)	1 以上
3 行目	点灯 1 色番号	下表参照
4 行目	点灯 2 色番号	下表参照 (省略した場合、0)

色番号	色
0	非点灯
1	赤
2	緑
3	黄
4	青
5	紫
6	水色
7	白
範囲外	非点灯

1 秒毎に黄色点滅
echo -e "1000\n1000\n3" > /tmp/.runled

1 秒毎に黄色/緑色交互点滅
echo -e "1000\n1000\n3\n2" > /tmp/.runled

12.2 モバイル回線向けモデム制御

本装置搭載のモデムモジュールの電源 ON/OFF や電波状態をチェックするツールを搭載しています。

コマンド名：atcmd

起動方法 1：atcmd [コマンド]

起動方法 2：atcmd [コマンド 1] [コマンド 2] [コマンド 3]

起動方法 3 ：atcmd -d [デバイスファイル] [コマンド]

起動方法 2 のように、コマンドを列挙して順次実行することが可能です。

また、起動方法 3 のように使用するデバイスファイルを指定実行することも可能です。

コマンド	機能	備考
PON	モデム電源 ON	
POFF	モデム電源 OFF	
PRST	モデム reboot	ソフトウェアリセット (一部、ハードウェアリセット)
HRST	モデム reboot	ハードウェアリセット
SMONI	電波強度取得	
CSQ	電波強度取得	LTE モジュール (NTT ドコモ/KDDI) 及び BWA モジュールでは、正しい電波強度取得は行えません。
SIND	基地局からの時刻取得	BX1 限定
CCLK	基地局からの時刻取得	BX3 限定
CCID	SIM の番号取得	
CTZU 1	タイムゾーン自動取得	BX1 及び EX/VX シリーズ LTE モジュール (NTT ドコモ) 以外のモデルのみ対応 ※スペースが入ります。
ATI	モデム型番取得	
CGSN	モデムシリアル番号取得	

コマンドは上記表の内容で指定します。

電源 ON、SIM 番号取得、電源 OFF
atcmd PON CCID POFF xxxxxxxxxxxxxx

電源 ON、電波強度取得、電源 OFF
atcmd PON CSQ POFF -86

WEB UI では常時電波状況取得によりデバイスファイルを占有しています。そのため、本コマンドは WEB UI にて使用していないデバイスファイルを指定し実行してください。尚、WEB UI を使用していない環境またはユーザー制御では本制約はありません。

尚、使用モジュールによって atcmd にて使用できるデバイスファイルが異なります。

BX1	
デバイスファイル	atcmd 使用可否
/dev/ttyMODEM0	使用可能
/dev/ttyMODEM1	使用不可
/dev/ttyMODEM2	使用不可
/dev/ttyMODEM3	使用可能
/dev/ttyMODEM4	使用可能
/dev/ttyMODEM5	使用不可
/dev/ttyMODEM6	使用不可

BX3	
デバイスファイル	atcmd 使用可否
/dev/ttyMODEM0	使用可能
/dev/ttyMODEM1	使用可能
/dev/ttyMODEM2	使用可能
/dev/ttyMODEM3	使用不可
/dev/ttyMODEM4	使用不可
/dev/ttyMODEM5	使用不可
/dev/ttyMODEM6	使用不可

BX5	
デバイスファイル	atcmd 使用可否
/dev/ttyMODEM0	使用可能
/dev/ttyMODEM1	使用不可
/dev/ttyMODEM2	使用不可
/dev/ttyMODEM3	使用可能
/dev/ttyMODEM4	使用不可

EX/VX シリーズ LTE/3G モジュール (ソフトバンク)	
デバイスファイル	atcmd 使用可否
/dev/ttyMODEM0	使用可能

EX/VX シリーズ LTE モジュール (KDDI)	
デバイスファイル	atcmd 使用可否
/dev/ttyMODEM0	使用可能

EX/VX シリーズ LTE モジュール (NTT ドコモ/KDDI)	
デバイスファイル	atcmd 使用可否
/dev/ttyMODEM0	使用可能
/dev/ttyMODEM1	使用不可
/dev/ttyMODEM2	使用不可
/dev/ttyMODEM3	使用可能
/dev/ttyMODEM4	使用不可

EX/VX シリーズ BWA モジュール	
デバイスファイル	atcmd 使用可否
/dev/ttyMODEM0	使用可能
/dev/ttyMODEM1	使用不可
/dev/ttyMODEM2	使用不可
/dev/ttyMODEM3	使用可能
/dev/ttyMODEM4	使用不可

EX1/3G モジュール (NTT ドコモ)	
デバイスファイル	atcmd 使用可否
/dev/ttyMODEM0	使用可能
/dev/ttyMODEM1	使用可能
/dev/ttyMODEM2	使用可能
/dev/ttyMODEM3	使用不可
/dev/ttyMODEM4	使用不可
/dev/ttyMODEM5	使用不可
/dev/ttyMODEM6	使用不可

EX1/3G モジュール (ソフトバンク)	
デバイスファイル	atcmd 使用可否
/dev/ttyMODEM0	使用可能
/dev/ttyMODEM1	使用可能
/dev/ttyMODEM2	使用可能
/dev/ttyMODEM3	使用不可
/dev/ttyMODEM4	使用不可
/dev/ttyMODEM5	使用不可
/dev/ttyMODEM6	使用不可

12.3 モバイル回線接続

モバイル回線モデムの制御は WEB UI にて行っております。

WEB UI 制御における必要時のみモバイル回線接続する方式 (オンデマンド接続) 及び WEB UI からモデムの制御を完全に排他したユーザー制御方式をサポートしています。

12.3.1 オンデマンド接続

WEB UI のネットワーク → 基本タブにアクセスし、モバイル回線の接続に必要な情報を設定します。

必要な情報は以下となります。
・APN(LTE モジュール (KDDI) 時は項目が有りません)
・ユーザー名
・パスワード
・認証方式
・自動接続 : "自動接続しない"を選択してください。
・通信確認用ホスト
・定期再接続設定
（・モバイル回線再接続時間）
・SMS コントロール (LTE モジュール (KDDI) 時は項目が有りません)
※ SMS コントロールは使用する場合にのみ設定してください。

保存ボタンを押し、再起動することでモバイル回線モデムへの反映は完了します。

また、モバイル回線のオンデマンド接続及び切断のコマンドは以下となります。尚、下記のコマンドラインにおける数字の"1"はオンデマンド接続を前提とするものです。"0"を指定した場合には、接続を保持する常時接続となります。

●モバイル回線の接続
/var/webui/scripts/mobile_control.sh con 1

●モバイル回線の切断
/var/webui/scripts/mobile_control.sh coff 1

上記のコマンドを組み合わせたシェルスクリプトのサンプルは以下です。

内容は以下となっております。

1, モバイル回線の接続。

2, DNS サーバに対して ping コマンドを実行。

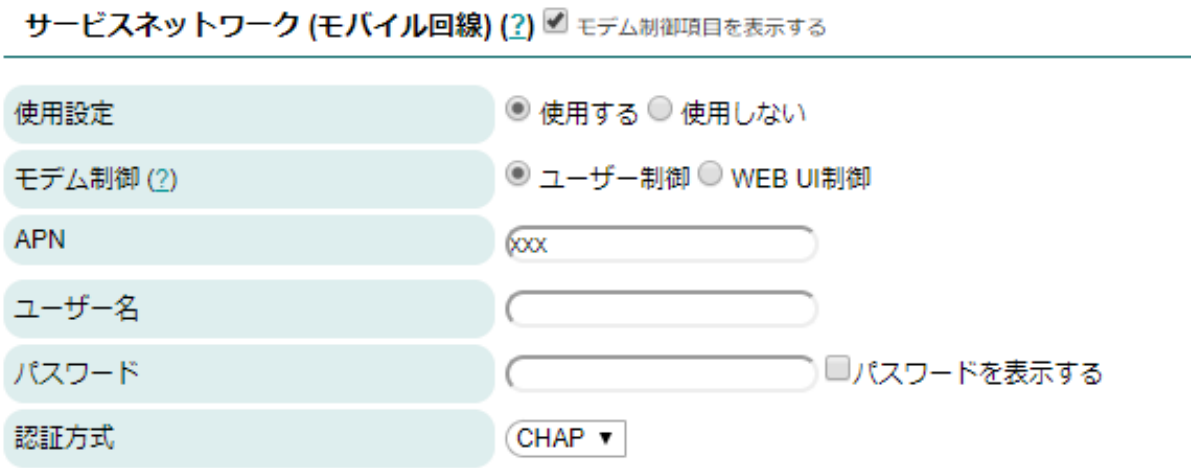
3, モバイル回線の切断。

```
#!/bin/bash
echo "#-----------------------------------------------------------#"
echo "# Connect (date)"
echo "#-----------------------------------------------------------#"
/var/webui/scripts/mobile_control.sh con 1
sleep 2
echo ""
echo "#-----------------------------------------------------------#"
echo "# Command Exec (date)"
echo "#-----------------------------------------------------------#"
ping -c 3 8.8.8.8
echo ""
echo "#-----------------------------------------------------------#"
echo "# Disonnect (date)"
echo "#-----------------------------------------------------------#"
/var/webui/scripts/mobile_control.sh coff 1
sleep 2
exit 0
```

12.3.2 モバイル回線モデムのユーザー制御

本項は 3G モジュール向けの説明となっております。

WEB UI のネットワーク → 基本タブにアクセスし、モバイル回線の接続に必要な情報を設定します。尚、LTE モジュールは本機能に対応しておりません。



本項目を実施する場合には、「モデム制御項目を表示する」にチェックをしてください。 設定に必要な情報は以下となります。 ・モデム制御：”ユーザー制御 ”を選択してください。 ・APN ・ユーザ名 ・パスワード ・認証方式

WEB UI にて設定した情報は以下のコマンドでモデムへ反映可能です。

●モデム設定反映コマンド
/var/webui/scripts/setapn.sh

モデム設定反映コマンドで使用しているコンフィグファイルは以下となります。

●モデム設定反映用コンフィグファイル
/var/webui/config/ppp0_device.sh /var/webui/upload_dir/modem.sh

設定情報をさらに変更した場合には、以下のファイルを作成し設定することにより情報を上書きすることが可能です。

●情報上書き用ファイル
/var/webui/upload_dir/user_modem.sh

●変数内容		
変数名	内容	備考
modem_ppp0_apn	APN	
modem_ppp0_user	ユーザー名	
modem_ppp0_pass	パスワード	
modem_ppp0_authtype	パスワード	PAP または CHAP
modem_ppp0_provier	プロバイダ名	pon/poff コマンドでの指定
DEVICE_CONNECT	モバイル回線接続用デバイスファイル	/dev/ttyMODEM[0-9]*となります。
DEVICE_SETTING	モバイル回線設定用デバイスファイル	/dev/ttyMODEM[0-9]*となります。

●設定サンプル
modem_ppp0_apn="iixxxx.jp" modem_ppp0_user="test@iixxxx" modem_ppp0_pass="xxxx" modem_ppp0_authtype="PAP" modem_ppp0_provier="usermobile"

以下の各コマンドにより、モバイル回線の接続及び切断が可能です。

●モバイル回線接続コマンド
pon <modem_ppp0_provier>

●モバイル回線切断コマンド
poff <modem_ppp0_provier>

※<modem_ppp0_provier>は変数で設定した物となります。user_modem.sh にて変更していない場合は"mobile"となります。

注釈: ●電波強度等の取得については、atmcd を使用してください。尚、回線接続しているデバイスファイルに対しては取得ができません。

注釈: ●モバイル回線の接続前にデフォルトゲートウェイが設定されている場合、モバイル回線接続時には反映されません。そのため、モバイル回線接続までにデフォルトゲートウェイを解除してください。また、モバイル回線切断後はデフォルトゲートウェイを解除している状態となります。そのため、インターネット環境へのアクセスを別途する場合等では、デフォルトゲートウェイを設定してください。

12.4 バックアップ

作りこみを行った本システムデータのバックアップは以下の方法にて行えます。

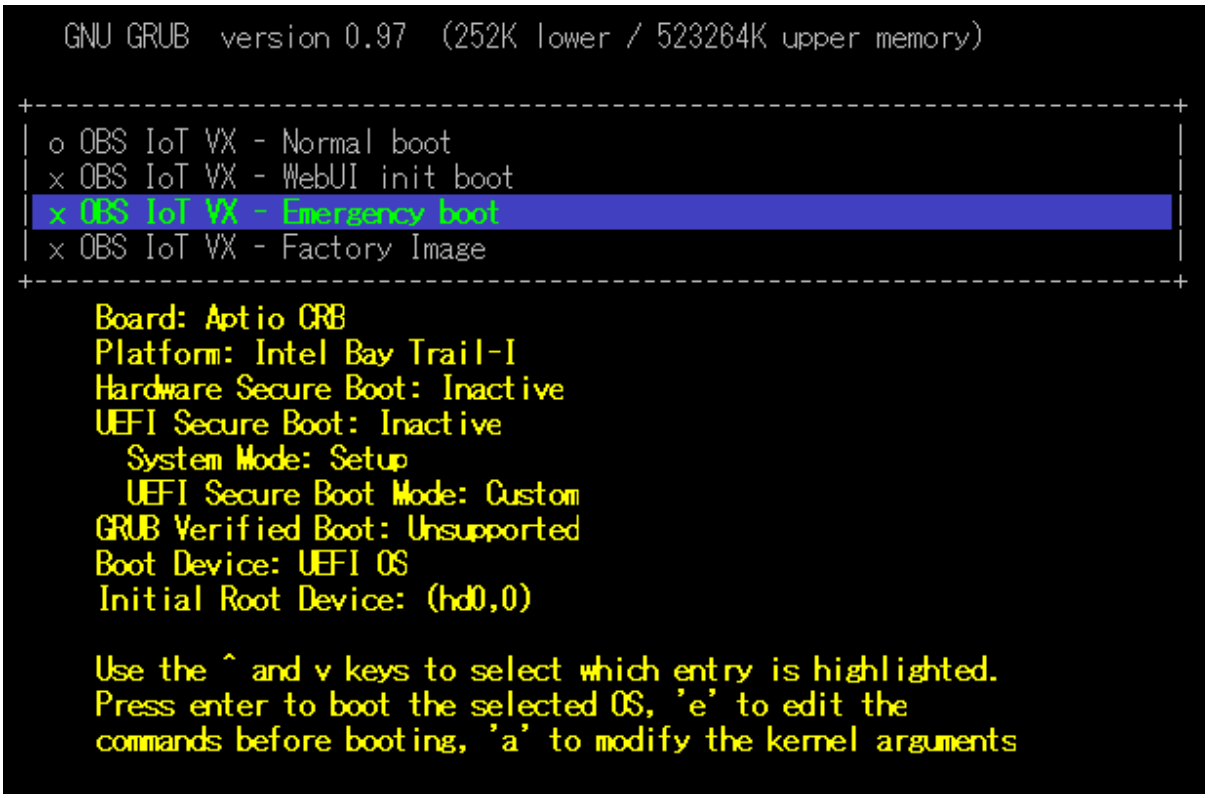
◆ OpenBlocks IoT VX シリーズ及び OpenBlocks IX9 の場合

1, 本体再起動

```
# sync
# reboot
```

2, エマージェンシーブートモードによる起動

起動時の GRUB 画面にて『Emergency boot』を選択して下さい。



3, 不要データの削除及びバックアップ

```
obsiot login: root
Password: 0BSI0T
# mount /dev/mmcblk0p2 /mnt
    ※/mnt 以下の HW 固有のファイル等は削除してください。以下は BT 情報を削除しています。
# rm -rf /mnt/var/lib/bluetooth/*
# cd /mnt
# tar --exclude=lost+found --exclude=<tgz ファイル名> -cpzf <tgz ファイル名> .
```

/mnt/<tgz ファイル>がバックアップしたファイルとなります。

注釈: /var/lib/bluetooth/ディレクトリ配下に HW 固有の情報が含まれています。また、他にも作りこみの際の中間ファイル等も削除した方が良いです。

◆ OpenBlocks IoT BX/EX シリーズの場合

1, リカバリーブート起動

OpenBlocks IoT BX/EX シリーズでは起動時に uboot 起動中にキー入力を行う事で、uboot プロンプトが表示されます。uboot プロンプトにて、リカバリー FW 起動用コマンドを実施し RAM ディスクモードにて起動することが行えます。

```
******************************
PSH KERNEL VERSION: b0182727
        WR: 20104000
******************************
SCU IPC: 0x800000d0 0xfffce92c
PSH miaHOB version: TNG.B0.VVBD.0000000c
microkernel built 23:15:13 Apr 24 2014
******* PSH loader *******
PCM page cache size = 192 KB
Cache Constraint = 0 Pages
Arming IPC driver ..
Adding page store pool ..
PagestoreAddr(IMR Start Address) = 0x04899000
pageStoreSize(IMR Size) = 0x00080000
*** Ready to receive application ***
U-Boot 2014.04 (Oct 14 2014 - 15:19:04)
        Watchdog enabled
DRAM: 980.6 MiB
MMC: tangier_sdhci: 0
In: serial
Out: serial
Err: serial
Hit any key to stop autoboot: 0              ※キー入力
boot > run bootRecovery
```

2, 不要データの削除及びバックアップ

```
obsiot login: root
Password: 0BSI0T
# mount /dev/mmcblk0p10 /mnt
      ※/mnt 以下の HW 固有のファイル等は削除してください。以下は BT 情報を削除しています。
# rm -rf /mnt/var/lib/bluetooth/*
# cd /mnt
# tar --exclude=lost+found --exclude=<tgz ファイル名> -cpzf <tgz ファイル名> .
```

/mnt/<tgz ファイル>がバックアップしたファイルとなります。

注釈: /var/lib/bluetooth/ディレクトリ配下に HW 固有の情報が含まれています。また、他にも作りこみの際の中間ファイル等も削除した方が良いです。

12.5 リストア

エマージェンシーブート (リカバリブートモード) にてボリュームラベルに「DEB_CONFIG」を設定したストレージ (ext2/ext3/vfat の USB メモリ等) に後述の応用の init.sh を用い、Primary パーティションにバックアップした<tgz ファイル>の展開を行いリストアします。

注釈: ・バックアップしたファイルの展開前に Primary となるパーティションストレージをフォーマットしてください。(OpenBlocks IoT VX2: /dev/mmcblk0p2、OpenBlocks IoT BX 及び EX シリーズ: /dev/mmcblk0p10、OpenBlocks IX9: /dev/mmcblk0p2 または/dev/sda2)

注釈: ・バックアップしたファイルと kernel-image は一致している必要があります。そのため、リストア対象の筐体の kernel-image は別途確認してください。

12.6 応用

エマージェンシーブート (リカバリブートモード) にてバックアップファイルの他、所定のファイル名でファイルをボリュームラベルに「DEB_CONFIG」内に用意することで、スクリプト実行などが可能です。いずれもファイルが存在する場合のみ有効になります。

・init.sh(sh スクリプト、改行コードは LF のみ)

KERNEL 起動中において overlayfs のマウント前に、実行を行います。

・post-init.sh(sh スクリプト、改行コードは LF のみ)

KERNEL 起動中において overlayfs のマウント完了後に、実行を行います。

尚、エマージェンシーブート (リカバリブートモード) では、RAMDISK モードで起動します。そのため、上記のマウント作業については特に考慮する必要はありません。

※本製品では systemd を採用しております。そのため、KERNEL 起動中の段階では systemd が起動していないため一部コマンド (poweroff/reboot コマンド等) が使用できません。これらのコマンド等を使用する場合には、処理したいコマンド等をバックグラウンドで起動すべき daemon 等が起動完了後に実行してください。

12.7 推奨使用デバイスファイル

Linux では各デバイスファイル等は認識順に名前がアサインされます。

そのため、デバイス自体の電源 ON/OFF 状況によってデバイスファイルが異なる恐れがあります。

モデル毎に自動でデバイスファイルのリンクを張りますので、デバイスファイルへのアクセスを行う場合には以下の推奨デバイスファイルを使用してください。

● OpenBlocks IoT VX シリーズの場合	
推奨デバイスファイル	対象デバイス
/dev/ttyRS485	RS-485 用デバイスファイル
/dev/ttyEX1	拡張スロット 1 用デバイスファイル (LoRaWAN 等)
/dev/ttyEX2	拡張スロット 2 用デバイスファイル (EnOcean 等)
/dev/S4	RS-232C 用デバイスファイル (VX1 のみ)

● OpenBlocks IoT EX1 の場合	
推奨デバイスファイル	対象デバイス
/dev/ttyEX1	拡張スロット 1 用デバイスファイル (LoRaWAN 等)
/dev/ttyEX2	拡張スロット 2 用デバイスファイル (EnOcean 等)
/dev/MFD1	RS-232C または RS-485 用デバイスファイル

● OpenBlocks IoT EX1G の場合	
推奨デバイスファイル	対象デバイス
/dev/ttyRS485	RS-485 用デバイスファイル
/dev/ttyEX1	拡張スロット 1 用デバイスファイル (LoRaWAN 等)
/dev/ttyEX2	拡張スロット 2 用デバイスファイル (EnOcean 等)
/dev/MFD1	RS-232C 用デバイスファイル

● OpenBlocks IX9 の場合	
推奨デバイスファイル	対象デバイス
/dev/S4	RS-232C ポート 0 用デバイスファイル
/dev/S5	RS-232C ポート 1 用デバイスファイル

12.8 Factory Reset(工場出荷状態への切り替え)

OpenBlocks IoT VX シリーズ及び IX9 にてストレージ領域へパッケージの追加や重要データの削除等を実施してしまい、工場出荷状態に戻したい場合、GRUB メニューの「Factory Image」を選択することで工場出荷状態へ戻すことが出来ます。

工場出荷状態に戻した場合には、設定したデータ等は削除されますのでご注意ください。

```
  GNU GRUB  version 0.97  (252K lower / 523264K upper memory)

+-------------------------------------------------------------------------+
| o OBS IoT VX - Normal boot                                              |
| x OBS IoT VX - WebUI init boot                                          |
| x OBS IoT VX - Emergency boot                                           |
| x OBS IoT VX - Factory Image                                            |
+-------------------------------------------------------------------------+
     Board: Aptio CRB
     Platform: Intel Bay Trail-I
     Hardware Secure Boot: Inactive
     UEFI Secure Boot: Inactive
       System Mode: Setup
       UEFI Secure Boot Mode: Custom
     GRUB Verified Boot: Unsupported
     Boot Device: UEFI OS
     Initial Root Device: (hd0,0)

     Use the ^ and v keys to select which entry is highlighted.
     Press enter to boot the selected OS, 'e' to edit the
     commands before booting, 'a' to modify the kernel arguments
```

また、OpenBlocks IoT BX/EX シリーズにて工場出荷状態に戻したい場合には弊社製品 HP の『ドキュメント』→『その他』→『ファクトリーリセット』をご確認し、作業を実施してください。

12.9 リカバリー起動

各モデルにて通常起動で用いている FW データやストレージデータが破損した場合、リカバリー用の FW にて起動することが出来ます。

OpenBlocks IX9 及び OpenBlocks IoT VX シリーズにて GRUB メニューの「Emergency boot」を選択することで、RAM Disk モードにて起動することが行えます。

本起動によるコンソールによるログインアカウント及びパスワードは"root"/"0BSI0T"となります。(パスワードの 0 は数字の 0 となります。)

```
   GNU GRUB  version 0.97  (252K lower / 523264K upper memory)

+-------------------------------------------------------------------------+
| o OBS IoT VX - Normal boot                                              |
| x OBS IoT VX - WebUI init boot                                          |
| x OBS IoT VX - Emergency boot                                           |
| x OBS IoT VX - Factory Image                                            |
+-------------------------------------------------------------------------+
     Board: Aptio CRB
     Platform: Intel Bay Trail-I
     Hardware Secure Boot: Inactive
     UEFI Secure Boot: Inactive
       System Mode: Setup
       UEFI Secure Boot Mode: Custom
     GRUB Verified Boot: Unsupported
     Boot Device: UEFI OS
     Initial Root Device: (hd0,0)

     Use the ^ and v keys to select which entry is highlighted.
     Press enter to boot the selected OS, 'e' to edit the
     commands before booting, 'a' to modify the kernel arguments
```

OpenBlocks IoT BX/EX シリーズでは起動時に uboot 起動中にキー入力を行う事で、uboot プロンプトが表示されます。

uboot プロンプトにて、リカバリー FW 起動用コマンドを実施し RAM ディスクモードにて起動することが行えます。

リカバリー FW 起動後に、kernel-image データを再インストールやストレージデータの復旧等を実施してください。

```
******************************
PSH KERNEL VERSION: b0182727
        WR: 20104000
******************************
SCU IPC: 0x800000d0 0xfffce92c
PSH miaHOB version: TNG.B0.VVBD.0000000c
microkernel built 23:15:13 Apr 24 2014
******* PSH loader *******
PCM page cache size = 192 KB
Cache Constraint = 0 Pages
Arming IPC driver ..
Adding page store pool ..
PagestoreAddr(IMR Start Address) = 0x04899000
pageStoreSize(IMR Size) = 0x00080000
*** Ready to receive application ***
U-Boot 2014.04 (Oct 14 2014 - 15:19:04)
        Watchdog enabled
DRAM: 980.6 MiB
MMC: tangier_sdhci: 0
In: serial
Out: serial
Err: serial
Hit any key to stop autoboot: 0            ※キー入力
boot > run bootRecovery
```

尚、WEB UI を用いている場合通常起動時においてコマンドラインにより以下のコマンドを実行することで次回以降の起動をリカバリーモード起動することが行えます。また、同様にリカバリーモード時から次回以降の起動を通常起動とするコマンドについても以下となります。

通常起動からリカバリーモード起動への切り替えコマンド
/var/webui/bin/bxex_uboot_runrecovery.sh

リカバリーモード起動から通常起動への切り替えコマンド
/usr/sbin/fw_setenv bootcmd 'setenv firm_part 0:7;run bootDebian;'

注釈: 通常起動からリカバリーモードへの切り替えコマンドを実施した場合、次回以降の起動は常にリカバリーモードとなります。そのため、実配置している OpenBlocks では実行しないでください。

12.10 クロス開発環境の構築

OpenBlocks シリーズのファームウェアの作成方法を、以下のページに用意しています。開発環境の構築を行う方はご確認ください。

https://github.com/plathome/debian_based_firmware

12.11 WEB UI 拡張機能

WEB UI 上にて、拡張機能として以下を用意しております。

12.11.1 WEB UI への機能拡張インストール

出荷直後状態の本筐体では、ネットワーク設定等を設定するソフトウェアのみ組み込まれています。IoT Gateway として使用する場合や Node-RED 等をインストールする等の機能を拡張することが行えます。インストール方法については、OpenBlocks シリーズ向け WEB UI セットアップガイドに記載しておりますので、そちらをご参照ください。

12.11.2 スクリプト編集

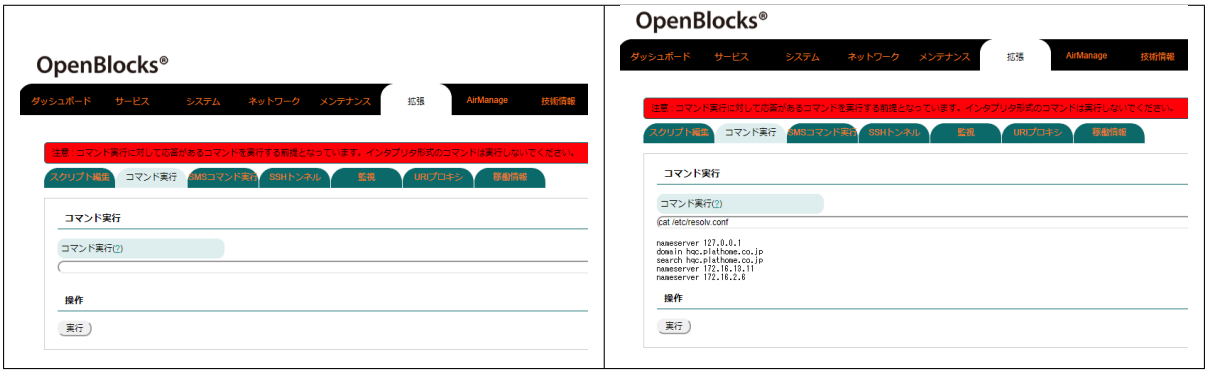
「拡張」→「スクリプト編集」タブにて、以下のスクリプトを WEB UI 上から作成・編集することが出来ます。

- 起動スクリプト
- 終了スクリプト
- ユーザー定義スクリプト 1〜5
- ユーザー定義スクリプト (button)
- 動体検知イベント

本機能はユーザー様の実装内容依存となりますので、ご注意ください。

#	スクリプト種類	実行タイミング
1	起動スクリプト	本製品の起動時における WEB UI の起動処理完了後に実行されます。
2	終了スクリプト	本製品の終了時における WEB UI の終了処理開始直後に実行されます。
3	ユーザー定義スクリプト 1〜5	通常処理中には実行されません。 SMS コントロール機能にて命令適用時に実施されます。
4	ユーザー定義スクリプト (button)	FUNC 機能割当機能にて設定した場合に、FUNC スイッチを押すことで実施されます。
5	動体検知イベント	カメラ機能における動体検知時の動画保存時に、イベントトリガーが有効となっている場合に実施されます。

12.11.3 コマンド実行



「拡張」→「コマンド実行」タブから 1 行程度のコマンド実行することが可能です。

コマンドを実行すると応答結果が表示されます。

本機能にてフォアグラウンドで永続稼働するようなコマンドを実行した場合、WEB 応答しなくなりますので注意してください。

12.11.4 SMS 送信

WEB UI を用いておりモデムモジュール (LTE モジュール (KDDI) を除く) が搭載され、SMS 送信可能な SIM が挿入されている場合、コマンドライン上から SMS を送信することが出来ます。

以下のコマンドにより、SMS データの雛形を作成します。

※雛形データを作成後、自動で送信されます。

LTE モジュール (NTT ドコモ) のモデルの場合 / 実行方法
/var/webui/bin/create_sms_um04.php <宛先電話番号> <本文>

LTE モジュール (NTT ドコモ) のモデルの場合 / 実行サンプル
/var/webui/bin/create_sms_um04.php 09012345678 "TEST MESSAGE"

LTE モジュール (NTT ドコモ) 以外のモデル / 実行方法
/var/webui/bin/create_sms.py <宛先電話番号> <本文>

LTE モジュール (NTT ドコモ) 以外のモデル / 実行サンプル
/var/webui/bin/create_sms.py 09012345678 "TEST MESSAGE"

12.11.5 LTE モジュール (KDDI) 運用切り替え

LTE モジュール (KDDI) はモジュール自体が SIM になる機能を持っています。以下のコマンドにより、モジュール自体が SIM の機能となる内部 SIM モード及び挿入されている SIM を参照する外部 SIM モードの切り替えが可能です。

内部 SIM モードへの切替コマンド
/var/webui/scripts/kym_set_mode.sh in

外部 SIM モードへの切替コマンド
/var/webui/scripts/kym_set_mode.sh out

12.11.6 WEB UI 自動外部ストレージマウント機能

WEB UI において特定のボリュームラベルの付いたデバイスが見つかった場合、自動でマウントされます。

WEB UI の機能等で保存先管理等を行う場合にご使用ください。

ボリュームラベル	マウント先	備考
WEBUI_STORAGE	/var/tmp/storage	ファイルシステムは NTFS のみをご利用いただけます。

12.11.7 ブートモード切替

OpenBlocks IoT VX2 及び OpenBlocks IX9 では「メンテナンス」→「ブートモード切替」タブから次回起動時のブートモードを設定することができます。

通常出荷状態では選択可能なブートモードは１つのみとなっています。

※/dev/mmcblk0p1/EFI/boot/bootx64.conf ファイル部の一部を書き換えることによって、この部分に表示することができますが、選択可能なブートモードでは工場出荷状態へ戻す等の実運用では危険な内容が含まれています。

そのため、内容を判断の上表示追加等を行ってください。

※ OpenBlocks IX9 の SSD ブート時は/dev/sda1/EFI/boot/bootx64.conf となります。

12.11.8 GPS 情報について

WEB UI を用いており LTE モジュール (NTT ドコモ/KDDI) 及び BWA モジュール搭載品において、GPS の使用設定を有効にし SIM が挿入されている場合、最終取得時点の位置情報を JSON 形式にて以下のファイルに書き込みを行っております。

位置情報を活用したい場合にはこちらのファイルをご使用ください。

GPS 情報 JSON ファイル
/tmp/.gps_posi.json

また、GPS の位置情報取得の際に上記ファイル内容を Node-RED の Unix ドメインソケットに対して書き込みを行います。

GPS 情報書き込み Unix ドメインソケット (Abstract ソケット)
\0/tmp/node-red-gpsin.sock

Node-RED にて GPS 情報を JSON へ追加する手法については、『Nod-RED スターターガイド』をご確認ください。

12.11.9 DHCP における DGW、DNS について

OpenBlocks シリーズの WEB UI 機能にて、使用するネットワークインターフェースの IP アドレス取得設定にて DHCP を設定している場合、DGW(デフォルトゲートウェイ) 及び DNS(DNS サーバー) 情報は DHCP で設定された情報が反映されます。

そのため、通常では DHCP サーバー側にてデフォルトゲートウェイ及び DNS サーバー情報が設定されている為問題ありません。しかしセキュリティの関係上、デフォルトゲートウェイまたは DNS サーバー情報を配布しない DHCP サーバーのネットワークを運用しているケースがあります。このような場合、以下のファイルを対象フォーマットで用意することで DHCP 環境において強制的にデフォルトゲートウェイ及び DNS サーバーを設定することができます。

尚、上記における強制的に付与するデフォルトゲートウェイは DHCP における IP 付与時に設定されます。また、DNS サーバー設定に関してはローカルネットワークのインターフェースの Up 時 (DHCP) に適用されます。

対象ファイル
/var/webui/upload_dir/force_network.sh

ファイルフォーマット
force_network_defaultroute=＜デフォルトゲートウェイ IP アドレス＞ force_network_nameservers="＜ DNS サーバー IP アドレス 1 ＞ ＜ DNS サーバー IP アドレス 2 ＞"

ファイルサンプル
force_network_defaultroute=192.168.130.1 force_network_nameservers= ”127.0.0.1 192.168.130.1 ”

12.11.10 Docker について

OpenBlocks IoT VX2 及び OpenBlocks IX9 では Docker の動作が可能です。

WEB UI の拡張機能追加により Docker をインストールすることができます。

また、以下のコマンドでインターネット環境からインストールすることが可能です。

Moby インストール
apt-get update # apt-get -y install moby-ce

Docker インストール
apt-get update # apt-get -y install docker-ce

現状では、Docker コンテナを WEB UI から制御する機能はありません。コンテナ操作を行う場合には、コマンドラインにて実施してください。また、本番環境に Docker を用いる場合には、コンテナ起動・停止等のコマンドに関しては Docker 公式ページを参照してください。

第13章　製品仕様

13.1 OpenBlocks IoT VX1 本体仕様

項目		内容
モデル		OBSVX1
型番		OBSVX1
CPU	モデル	Intel Atom E3805
	動作速度	1.3GHz (デュアルコア)
	内蔵 2 次キャッシュ	1024kB/Core
メインメモリ	オンボード	1GB(64bit bus DDR3L)
内蔵ストレージ		8GB(eMMC)
追加ストレージ		MMC スロット x 1
無線インターフェース		BT 4.0+2.1 EDR
		WLAN(IEEE802.11a/b/g/n/ac) ※ 4
SIM インターフェース		通信用 miniSIM カードスロット
有線インターフェース	USB(HOST)	3.0 x 1 (type-A) ※ 2
	USB(Console)	Micro USB(type-B) x 1
	Ethernet	10Base-T/100BASE-TX/1000BASE-T x 1
	RS-232C	RJ45 x 1
	RS-485	半 2 重 x 1 (使用可能電線範囲:AWG28～AWG22)
寸法		91.9mm(W) x 114.8mm (D) x 25mm(H) (突起部含まず)
重量		160g
電源		DC-Jack：DC4.75～5.25V
		Wide DC 給電時 : DC10～48V ※ 3
消費電力	アイドル時	5.1W
	高負荷時	8.2W
MTBF		435,613h
EMC 規格		VCCI class A
省エネ法に基づくエネルギー		区分：H
消費効率 [単位：W/GTOPS] ※ 1		0.52
環境保護		RoHS 指令適合
認証		JATE / TELEC
RTC バックアップ時間		10 年
出荷時 OS		Debian GNU/Linux

※ 1 : エネルギー消費効率とは省エネ法で定める測定方法により測定した消費電力を省エネ法で定める複

合理論性能で除したものです。

※ 2 : サポートするケーブル長は 3m 未満です。

※ 3 : 本機能をご使用いただく場合、外付けノイズフィルター (SNR-10-223-T (COSEL)) または同等のノイズフィルターの接続が必要です。

※ 4 : 802.11ac のアクセスポイント機能はサポートいたしておりません。

13.2 OpenBlocks IoT VX2 本体仕様

項目		内容	
モデル		OBSVX2	
型番		OBSVX2	OBSVX2/N
CPU	モデル	Intel Atom E3805	
	動作速度	1.3GHz (デュアルコア)	
	内蔵 2 次キャッシュ	1024kB/Core	
メインメモリ	オンボード	2GB(64bit bus DDR3L)	
内蔵ストレージ		32GB(eMMC)	
追加ストレージ		MMC スロット x 1	
無線インターフェース		BT 4.0+2.1 EDR	
		WLAN(IEEE802.11a/b/g/n/ac) ※ 4	
SIM インターフェース		通信用 miniSIM カードスロット	通信用 nanoSIM カードスロット
有線インターフェース	USB(HOST)	3.0 x 1 (type-A) ※ 2	
	USB(Console)	Micro USB(type-B) x 1	
	Ethernet	10Base-T/100BASE-TX/1000BASE-T x 2	
	RS-485	半 2 重 x 1 (使用可能電線範囲:AWG28〜AWG22)	
寸法		91.9mm(W) x 114.8mm (D) x 25mm(H) (突起部含まず)	
重量		160g	
電源		DC-Jack：DC4.75〜5.25V	
		Wide DC 給電時 : DC10〜48V ※ 3	
消費電力	アイドル時	5.5W	
	高負荷時	9.0W	
MTBF		435,613h	
EMC 規格		VCCI class A	
省エネ法に基づくエネルギー消費効率 [単位：W/GTOPS] ※ 1		区分：H	
		0.52	
環境保護		RoHS 指令適合	
認証		JATE / TELEC	
RTC バックアップ時間		10 年	
出荷時 OS		Debian GNU/Linux	

※ 1 : エネルギー消費効率とは省エネ法で定める測定方法により測定した消費電力を省エネ法で定める複

合理論性能で除したものです。

※ 2 : サポートするケーブル長は 3m 未満です。

※ 3 : 本機能をご使用いただく場合、外付けノイズフィルター (SNR-10-223-T (COSEL)) または同等のノイズフィルターの接続が必要です。

※ 4 : 802.11ac のアクセスポイント機能はサポートいたしておりません。

13.3 OpenBlocks IoT VX シリーズオプション

13.3.1 LTE/3G モジュール (ソフトバンク)

項目		内容
対応周波数または Band	GSM	850/900/1800/1900MHz
	W-CDMA	850/900/1900/2100MHz
	LTE	(1)/(3)/(5)/(7)/(8)/(20)
データ通信速度		下り：150Mbps , 上り：50Mbps (理論値)
制御方法		AT コマンド・特殊コマンド
消費電力	アイドル時	0.05W (平均電力)
	高負荷時	2.5W (平均電力)
認証		JATE/TELEC
MTBF		97,808h@25 ℃

※ 1 : () 内は Band

13.3.2 LTE モジュール (KDDI)

項目		内容
対応周波数	受信	860.0MHz〜875.0MHz
	送信	815.0MHz〜830.0MHz
データ通信速度		下り：75Mbps , 上り：25Mbps (理論値)
制御方法		AT コマンド・特殊コマンド
認証		JATE/TELEC
消費電力	アイドル時	0.16W (平均電力)
	高負荷時	2.4W (平均電力)

13.3.3 LTE モジュール (NTT ドコモ/KDDI)

項目		内容
対応周波数または Band	GSM	900/1800MHz
	W-CDMA	800(6)/900(8)/2100MHz
	LTE	800(18)/800(19)/900(8)/1800(3)/2100(1)/2600(41)MHz ※ 1
	GNSS	GPS/GLONASS
データ通信速度		下り：100Mbps , 上り：50Mbps (理論値)
制御方法		AT コマンド・特殊コマンド
認証		JATE[150051003] / TELEC[003-150053]
消費電力	アイドル時	0.02W (平均電力)
	高負荷時	2.9W (平均電力)
MTBF		500,000h @25 ℃

※ 1 : () 内は Band

13.3.4 BWA モジュール

項目		内容
対応周波数または Band	GSM	900/1800MHz
	W-CDMA	800(6)/900(8)/2100MHz
	LTE	800(18)/800(19)/900(8)/1800(3)/2100(1)/2600(41)MHz ※ 1
	GNSS	GPS/GLONASS
データ通信速度		下り：100Mbps , 上り：50Mbps (理論値)
制御方法		AT コマンド・特殊コマンド
認証		JATE[150051003] / TELEC[003-150053]
消費電力	アイドル時	0.02W (平均電力)
	高負荷時	2.9W (平均電力)
MTBF		500,000h @25 ℃

※ 1 : () 内は Band

13.3.5 EnOcean モジュール

項目		内容
周波数		928.35MHz
データ通信速度		125kbps
HOST I/F		UART
認証		TELEC
消費電力	受信時	0.13W
	送信時	0.11W
MTBF		862,664h @25 ℃

13.3.6 Wi-SUN モジュール

項目		内容
周波数		922.5〜927.9MHz(28ch)
データ通信速度		100kbps
HOST I/F		UART
認証		TELEC
消費電力	受信時	0.1W
	送信時	0.15W
MTBF		559,409h @25 ℃

13.3.7 LoRaWAN モジュール

項目		内容
周波数		920.6〜928.0MHz
データ通信速度		11kbps
HOST I/F		UART
認証		TELEC
消費電力	受信時	0.05W
	送信時	0.12W
MTBF		4,076,190h @25 ℃

13.4 OpenBlocks IoT VX2 オプション

13.4.1 内蔵バッテリー

項目	仕様	備考
HOST I/F	I2C	充電制御、バッテリーモジュール電圧検出
公称電圧	4.8V	
定格容量	500mAh	
充電電流	急速充電:150mA , トリクル充電:20mA	
充電時間	急速充電：2h , トリクル充電：13h	バックアップ機能は急速充電終了後に使用可能
放電時間	装置負荷による	(参考) 定格容量÷消費電流× 0.5 [h]
放電終止電圧	4.0 〜 4.4V	
使用温湿度範囲	0 〜 +40 ℃ / 45 〜 85%	
保存温湿度範囲	-20 〜 +35 ℃ / 45 〜 85%	1 年以内の場合
保護部品	サーマルプロテクター	電池パックに内蔵
寿命 (充放電回数)	使用条件による	(参考) 500 回または 2〜3 年

※非工場出荷オプションとなります。

注釈: 必ず OpenBlocks IoT Family 用 バッテリーモジュール使用上の注意をお読みの上ご使用ください。

13.4.2 ミスター省エネモジュール

電源仕様		
項目	仕様	備考
電源電圧	5.0V,3.3V	OpenBlocks IoT EX1 または VX2 側から供給
消費電流	最大 50mA	

シリアル通信仕様	
項目	仕様
通信方式	USB2.0 準拠
転送レート	1.2k, 2.4k, 4.8k, 9.6k, 19.2k, 38.4k, 57.6k, 115.2kbps
クロックドリフト	±100ppm/sec 以内
シリアルフレーム	スタートビット：1, データ長：7/8bit, パリティ：偶数/奇数/無し, ストップビット：1/2, フロー制御：無し
転送モード	アスキー (ASCII) モード
プロトコル	センサネットプロトコル（SNP）

無線仕様		
項目	仕様	備考
無線部方式	特定小電力無線	ARIB STD-T108 準拠
キャリア周波数	920.5 〜 929.70MHz	
変調方式	FSK	
伝送速度	100kbps	
設定可能チャネル	25-31,34-60,64-75（1mW） 25-31,34-60（20mW）	
選択可能チャネル数	46（1mW） 34（20mW）	
送信出力	1mW / 20mW	
通信距離	100m 以上（見込み）	屋外見通し（使用環境により異なります）

13.5 OpenBlocks IX9 本体仕様

項目		内容
モデル		OBSIX9
型番		OBSIX9
CPU	モデル	Intel Atom E3845
	動作速度	1.9GHz (クアッドコア)
	内蔵 2 次キャッシュ	1024kB/Core
メインメモリ	オンボード	8GB(Unbuffered DDR3L SDRAM SODIMM module w/o ECC)
内蔵ストレージ (オンボード)		8GB(eMMC 4.5)
グラフィックコントローラ		Intel HD Graphics for Intel Atom Processor Z3700 Series
セキュアブート		Yes
外部インターフェース	Ethernet	10BASE-T / 100BASE-TX / 1000BASE-T RJ45 × 4
	USB	3.0 x 2 (type-A) ※ 1
	RS-232C	RJ-45 x 2 (Rx, Tx, CTS, RTS) ※ 2
	Console	MicroUSB type-B
	Display	Mini DisplayPort(Single mode) ※ 3
内部インターフェース	M.2 Type 2280,key M	SATA 3.0Gbps
	M.2 Type 2280, key B	PCIe Gen2, USB3.0, SIM, I2C
寸法		101mm(W) x 142.1mm(D) x 41mm(H) ※ 4
重量		370g
電源		DC12V
消費電力	アイドル時	12.9W
	高負荷時	18.3W
MTBF		231,254h
EMC 規格		VCCI class A
省エネ法に基づくエネルギー消費効率 [単位：W/GTOPS]		区分：H
		0.62
環境保護		RoHS 指令適合
RTC バックアップ時間		10 年
出荷時 OS		Debian GNU/Linux

※ 1 : バスパワーは 2 ポートの合計が 1.5A を超えないこと。USB3.0 のケーブル長は 3m 以下。

※ 2 : RS-232C のケーブル長は 15m 以下。

※ 3 : DVI 出力や HDMI 出力を行う場合にはアクティブタイプの変換アダプタが必用。

※ 4 : ゴム足・突起部含まず。

13.6 OpenBlocks IoT BX1 本体仕様

項目		内容
モデル		OBSBX1
型番		OBSBX1
CPU	モデル	Intel Atom プロセッサー
	動作速度	500MHz (デュアルコア)
メインメモリ		1GB(LPDDR3)
内蔵ストレージ		4GB(eMMC)
無線インターフェース		3G(W-CDMA) ※ 1
		BT4.0
		WLAN(IEEE802.11a/b/g/n)
SIM インターフェース		通信用 miniSIM カードスロット
有線インターフェース		BX1 コネクタ標準添付：USB2.0(バスパワー・type-A)
寸法		41.6mm(W) x 96mm (D) x 11.3mm(H) (突起部含まず)
重量		45g
電源		DC 4.75〜48V(USB ケーブルによる給電時 DC4.75〜5.25V)
消費電力	アイドル時	3G 通信アクティブ時：0.8W / 3G 通信非アクティブ時：0.1W ※ 2
	高負荷時	4.6W
MTBF		224,858h
EMC 規格		VCCI class B 適合
省エネ法に基づくエネルギー消費効率 [単位：W/GTOPS] ※ 3		区分：H
		0.17
環境保護		RoHS 指令 PFOS 規制 省エネルギー法
認証		JATE / TELEC
RTC バックアップ時間		最大約 10 分
出荷時 OS		Debian GNU/Linux

※ 1 : NTT ドコモ FOMA 網に対応

※ 2 : Power Save = Auto 設定時

※ 3 : エネルギー消費効率とは省エネ法で定める測定方法により測定した消費電力を省エネ法で定める複合理論性能で除したものです

13.7 OpenBlocks IoT BX3 本体仕様

項目		内容
モデル		OBSBX3
型番		OBSBX3
CPU	モデル	Intel Atom プロセッサー
	動作速度	500MHz (デュアルコア)
メインメモリ		1GB(LPDDR3)
内蔵ストレージ		4GB(eMMC)
無線インターフェース		3G(W-CDMA) ※ 1
		BT4.0
		WLAN(IEEE802.11a/b/g/n)
SIM インターフェース		通信用 miniSIM カードスロット
有線インターフェース		BX1 コネクタ標準添付：USB2.0(バスパワー・type-A)
寸法		41.6mm(W) x 96mm (D) x 11.3mm(H) (突起部含まず)
重量		45g
電源		DC 4.75〜48V(USB ケーブルによる給電時 DC4.75〜5.25V)
消費電力	アイドル時	1.3W
	高負荷時	4.6W
MTBF		224,858h
EMC 規格		VCCI class B 適合
省エネ法に基づくエネルギー消費効率 [単位：W/GTOPS] ※ 2		区分：H
		0.17
環境保護		RoHS 指令 PFOS 規制 省エネルギー法
認証		JATE / TELEC
RTC バックアップ時間		最大約 10 分
出荷時 OS		Debian GNU/Linux

※ 1 : SoftBank 3G 網に対応

※ 2 : エネルギー消費効率とは省エネ法で定める測定方法により測定した消費電力を省エネ法で定める複合理論性能で除したものです

13.8 OpenBlocks IoT BX5 本体仕様

項目		内容
モデル		OBSBX5
型番		OBSBX5
CPU	モデル	Intel Atom プロセッサー
	動作速度	500MHz (デュアルコア)
メインメモリ		1GB(LPDDR3)
内蔵ストレージ		4GB(eMMC)
無線インターフェース		LTE(W-CDMA 含む) ※ 1
		BT4.0
		WLAN(IEEE802.11a/b/g/n)
SIM インターフェース		通信用 miniSIM カードスロット
有線インターフェース		BX1 コネクタ標準添付：USB2.0(バスパワー・type-A)
寸法		41.6mm(W) x 96mm (D) x 11.3mm(H) (突起部含まず)
重量		45g
電源		DC 4.75〜48V(USB ケーブルによる給電時 DC4.75〜5.25V)
消費電力	アイドル時	1.4W
	高負荷時	4.6W
MTBF		273,475h
EMC 規格		VCCI class B 適合
省エネ法に基づくエネルギー消費効率 [単位：W/GTOPS] ※ 2		区分：H
		0.18
環境保護		RoHS 指令 PFOS 規制 省エネルギー法
認証		JATE / TELEC
RTC バックアップ時間		最大約 10 分
出荷時 OS		Debian GNU/Linux

※ 1 : NTT ドコモ/KDDI/地域 BWA 網に対応

※ 2 : エネルギー消費効率とは省エネ法で定める測定方法により測定した消費電力を省エネ法で定める複合理論性能で除したものです

13.9 OpenBlocks IoT BX0 本体仕様

項目		内容
モデル		OBSBX5
型番		OBSBX5
CPU	モデル	Intel Atom プロセッサー
	動作速度	500MHz (デュアルコア)
メインメモリ		1GB(LPDDR3)
内蔵ストレージ		4GB(eMMC)
無線インターフェース		BT4.0
		WLAN(IEEE802.11a/b/g/n)
SIM インターフェース		通信用 miniSIM カードスロット
有線インターフェース		BX1 コネクタ標準添付：USB2.0(バスパワー・type-A , Ethernet 二股)
寸法		41.6mm(W) x 96mm (D) x 11.3mm(H) (突起部含まず)
重量		45g
電源		DC 4.75〜48V(USB ケーブルによる給電時 DC4.75〜5.25V) PoE 受電 (DC48V) TypeA(1-2,3-6)
消費電力	アイドル時	1.9W
	高負荷時	2.4W
MTBF		479,898h
EMC 規格		VCCI class B 適合
省エネ法に基づくエネルギー消費効率 [単位：W/GTOPS] ※ 1		区分：H
		0.24
環境保護		RoHS 指令 PFOS 規制 省エネルギー法
認証		TELEC
RTC バックアップ時間		約 16 年
出荷時 OS		Debian GNU/Linux

※ 1 : エネルギー消費効率とは省エネ法で定める測定方法により測定した消費電力を省エネ法で定める複合理論性能で除したものです

13.10 OpenBlocks IoT EX1 本体仕様

<table>
<tr><td colspan="2">項目</td><td colspan="3">内容</td></tr>
<tr><td colspan="2">モデル</td><td>OBSEX1</td><td colspan="2">OBSEX1G</td></tr>
<tr><td colspan="2">型番</td><td>OBSEX1</td><td>OBSEX1G</td><td>OBSEX1G/N</td></tr>
<tr><td rowspan="2">CPU</td><td>モデル</td><td colspan="3">Intel Atom E3805</td></tr>
<tr><td>動作速度</td><td colspan="3">1.3GHz (デュアルコア)</td></tr>
<tr><td colspan="2">メインメモリ</td><td colspan="3">1GB(LPDDR3)</td></tr>
<tr><td colspan="2">内蔵ストレージ</td><td colspan="3">4GB(eMMC)</td></tr>
<tr><td colspan="2">追加ストレージ</td><td colspan="3">MMC スロット x 1</td></tr>
<tr><td colspan="2" rowspan="2">無線インターフェース</td><td colspan="3">BT4.0</td></tr>
<tr><td colspan="3">WLAN(IEEE802.11a/b/g/n)</td></tr>
<tr><td colspan="2">SIM インターフェース</td><td colspan="2">通信用 miniSIM カードスロット</td><td>通信用 nanoSIM カードスロット</td></tr>
<tr><td rowspan="4">有線インターフェース</td><td>USB(HOST)</td><td colspan="3">2.0 x 1 (type-A)</td></tr>
<tr><td>USB(Console)</td><td colspan="3">Micro USB(type-B) x 1 ※バスパワー給電可</td></tr>
<tr><td>Ethernet</td><td colspan="3">10Base-T/100BASE-TX x 1</td></tr>
<tr><td>RS-232C/RS-485</td><td>RS-232C x 1：RJ45 コネクタ / RS-485 x 1：ターミナルブロック (排他)</td><td colspan="2">RS-232C x 1：RJ45 コネクタ / RS-485 x 1：ターミナルブロック</td></tr>
<tr><td colspan="2">寸法</td><td colspan="3">91.9mm(W) x 114.8mm (D) x 25mm(H) (突起部含まず)</td></tr>
<tr><td colspan="2">重量</td><td colspan="3">135g</td></tr>
<tr><td colspan="2">電源</td><td>DC-Jack：DC4.75〜5.25V
USB ケーブル給電時 : DC4.75〜5.25V</td><td colspan="2">DC-Jack：DC4.75〜5.25V
USB ケーブル給電時 : DC4.75〜5.25V
XA コネクタ給電時 : DC7〜48V</td></tr>
<tr><td rowspan="2">消費電力</td><td>アイドル時</td><td colspan="3">1.8W</td></tr>
<tr><td>高負荷時</td><td colspan="3">3.3W</td></tr>
<tr><td colspan="2">MTBF</td><td>485,265h</td><td colspan="2">468,231h</td></tr>
<tr><td colspan="2">EMC 規格</td><td></td><td colspan="2">VCCI class B 適合</td></tr>
<tr><td colspan="2" rowspan="2">省エネ法に基づくエネルギー消費効率 [単位：W/GTOPS] ※ 1</td><td colspan="3">区分：H</td></tr>
<tr><td colspan="3">0.23</td></tr>
<tr><td colspan="2">環境保護</td><td colspan="3">RoHS 指令 PFOS 規制 省エネルギー法</td></tr>
<tr><td colspan="2">認証</td><td colspan="3">TELEC</td></tr>
<tr><td colspan="2">RTC バックアップ時間</td><td>最大 250 日</td><td colspan="2">約 10 年</td></tr>
<tr><td colspan="2">出荷時 OS</td><td colspan="3">Debian GNU/Linux</td></tr>
</table>

※ 1 : エネルギー消費効率とは省エネ法で定める測定方法により測定した消費電力を省エネ法で定める複合理論性能で除したものです

13.11 OpenBlocks IoT EX1 オプション

13.11.1 3G モジュール (NTT ドコモ)

項目		内容
対応周波数	GSM/GPRS/EDGE	850/900/1800/1900MHz
	W-CDMA(UMTS/HSPA+)	Five band 800/850/900/1900/2100MHz
データ通信速度		下り：7.2Mbps , 上り：5.7Mbps (理論値)
制御方法		AT コマンド
認証		JATE/TELEC
電源電圧		DC 3.3〜4.4V
消費電力	アイドル時	0.01W (平均電力)
	高負荷時	2.6W (平均電力)

13.11.2 3G モジュール (ソフトバンク)

項目		内容
対応周波数	GSM/GPRS/EDGE	850/900/1800/1900MHz
	W-CDMA(UMTS/HSPA+)	Five band 800/850/900/1900/2100MHz
データ通信速度		下り：7.2Mbps , 上り：5.7Mbps (理論値)
制御方法		AT コマンド
認証		JATE/TELEC
電源電圧		DC 3.3〜4.4V
消費電力	アイドル時	0.01W (平均電力)
	高負荷時	2.6W (平均電力)

13.11.3 LTE/3G モジュール (ソフトバンク)

項目		内容
対応周波数または Band	GSM	850/900/1800/1900MHz
	W-CDMA	850/900/1900/2100MHz
	LTE	(1)/(3)/(5)/(7)/(8)/(20)
データ通信速度		下り：150Mbps , 上り：50Mbps (理論値)
制御方法		AT コマンド・特殊コマンド
消費電力	アイドル時	0.05W (平均電力)
	高負荷時	2.5W (平均電力)
認証		JATE/TELEC
MTBF		97,808h@25 ℃

※ 1 : () 内は Band

13.11.4 LTE モジュール (KDDI)

項目		内容
対応周波数	受信	860.0MHz～875.0MHz
	送信	815.0MHz～830.0MHz
データ通信速度		下り：75Mbps , 上り：25Mbps (理論値)
制御方法		AT コマンド・特殊コマンド
認証		JATE/TELEC
消費電力	アイドル時	0.16W (平均電力)
	高負荷時	2.4W (平均電力)

13.11.5 LTE モジュール (NTT ドコモ/KDDI)

項目		内容
対応周波数または Band	GSM	900/1800MHz
	W-CDMA	800(6)/900(8)/2100MHz
	LTE	800(18)/800(19)/900(8)/1800(3)/2100(1)/2600(41)MHz ※ 1
	GNSS	GPS/GLONASS
データ通信速度		下り：100Mbps , 上り：50Mbps (理論値)
制御方法		AT コマンド・特殊コマンド
認証		JATE[150051003] / TELEC[003-150053]
消費電力	アイドル時	0.02W (平均電力)
	高負荷時	2.9W (平均電力)
MTBF		500,000h @25 ℃

※ 1 : () 内は Band

13.11.6 BWA モジュール

項目		内容
対応周波数または Band	GSM	900/1800MHz
	W-CDMA	800(6)/900(8)/2100MHz
	LTE	800(18)/800(19)/900(8)/1800(3)/2100(1)/2600(41)MHz ※ 1
	GNSS	GPS/GLONASS
データ通信速度		下り：100Mbps , 上り：50Mbps (理論値)
制御方法		AT コマンド・特殊コマンド
認証		JATE[150051003] / TELEC[003-150053]
消費電力	アイドル時	0.02W (平均電力)
	高負荷時	2.9W (平均電力)
MTBF		500,000h @25 ℃

※ 1 : () 内は Band

13.11.7 EnOcean モジュール

項目		内容
周波数		928.35MHz
データ通信速度		125kbps
HOST I/F		UART
認証		TELEC
消費電力	受信時	0.13W
	送信時	0.11W
MTBF		862,664h @25 ℃

13.11.8 Wi-SUN モジュール

項目		内容
周波数		922.5～927.9MHz(28ch)
データ通信速度		100kbps
HOST I/F		UART
認証		TELEC
消費電力	受信時	0.1W
	送信時	0.15W
MTBF		559,409h @25 ℃

13.11.9 LoRaWAN モジュール

項目		内容
周波数		920.6～928.0MHz
データ通信速度		11kbps
HOST I/F		UART
認証		TELEC
消費電力	受信時	0.05W
	送信時	0.12W
MTBF		4,076,190h @25 ℃

13.11.10 内蔵バッテリー

項目	仕様	備考
HOST I/F	I2C	充電制御、バッテリーモジュール電圧検出
公称電圧	4.8V	
定格容量	500mAh	
充電電流	急速充電:150mA , トリクル充電:20mA	
充電時間	急速充電：2h , トリクル充電：13h	バックアップ機能は急速充電終了後に使用可能
放電時間	装置負荷による	(参考) 定格容量÷消費電流× 0.5 [h]
放電終止電圧	4.0 〜 4.4V	
使用温湿度範囲	0 〜 +40 ℃ / 45 〜 85%	
保存温湿度範囲	-20 〜 +35 ℃ / 45 〜 85%	1 年以内の場合
保護部品	サーマルプロテクター	電池パックに内蔵
寿命 (充放電回数)	使用条件による	(参考) 500 回または 2〜3 年

※非工場出荷オプションとなります。

注釈: 必ず OpenBlocks IoT Family 用 バッテリーモジュール使用上の注意をお読みの上ご使用ください。

13.11.11 ミスター省エネモジュール

電源仕様		
項目	仕様	備考
電源電圧	5.0V,3.3V	OpenBlocks IoT EX1 または VX2 側から供給
消費電流	最大 50mA	

シリアル通信仕様	
項目	仕様
通信方式	USB2.0 準拠
転送レート	1.2k, 2.4k, 4.8k, 9.6k, 19.2k, 38.4k, 57.6k, 115.2kbps
クロックドリフト	±100ppm/sec 以内
シリアルフレーム	スタートビット：1, データ長：7/8bit, パリティ：偶数/奇数/無し, ストップビット：1/2, フロー制御：無し
転送モード	アスキー (ASCII) モード
プロトコル	センサネットプロトコル（SNP）

無線仕様		
項目	仕様	備考
無線部方式	特定小電力無線	ARIB STD-T108 準拠
キャリア周波数	920.5 〜 929.70MHz	
変調方式	FSK	
伝送速度	100kbps	
設定可能チャネル	25-31,34-60,64-75（1mW） 25-31,34-60（20mW）	
選択可能チャネル数	46（1mW） 34（20mW）	
送信出力	1mW / 20mW	
通信距離	100m 以上（見込み）	屋外見通し（使用環境により異なります）

第14章　注意事項及び補足

14.1 スクリプト処理による遅延処理等について

エマージェンシーブート時における init.sh 及び post-init.sh 等のスクリプトにおいてストレージへの書き込みを行った場合、書き込み完了前に次のコマンド処理へと実施される場合があります。そのため、明示的に sleep 及び sync コマンド等を実施してください。

14.2 使用ポート一覧

WEB UI 込みでの OpenBlocks シリーズでは以下のポートを使用及び使用する可能性があります。

サービス種類	ポート番号	補足
FTP	21	FTP インストール時
SSH	22	ポート番号変更可能
DNS	53	
DHCP	67	
NTP	123	
NetBIOS	137	Samba インストール時 (UDP)
NetBIOS	138	Samba インストール時 (UDP)
NetBIOS	139	Samba インストール時
Samba	445	Samba インストール時
Modbus	502	IoT データ制御インストール時
WEB UI(HTTP アクセス)	880	
ECHONET	3610	IoT データ制御/HVSMC 使用時
Node-RED	1880	Node-RED インストール時 (ポート番号変更可能)
Shell in a box(WEB SSH)	4200	
WEB UI(HTTPS アクセス)	4430	
SSH	50022	LTE/3G モジュール (ソフトバンク) / WAN 側のみ
WEB UI(HTTP アクセス)	50880	LTE/3G モジュール (ソフトバンク) / WAN 側のみ
WEB UI(HTTPS アクセス)	54430	LTE/3G モジュール (ソフトバンク) / WAN 側のみ
WEB UI 独自サービス	63003	

14.3 OpenBlocks IX9 の Mini DisplayPort について

OpenBlocks IX9 には X Window System はインストールされていません。そのため、Mini DisplayPort からの出力内容はコンソール画面のみとなります。

Mini DisplayPort にてデスクトップ環境を使用したい場合には、tasksel コマンドにてデスクトップ環境のインストールを行ってください。

尚、デスクトップ環境のインストールにて約 1GB 程度のストレージを消費します。そのため、ストレージに空き容量が十分に存在することを確認してください。

第V部

データハンドリングガイド

第15章　はじめに

OpenBlocks シリーズにて用いているデータハンドリング機能について解説しています。

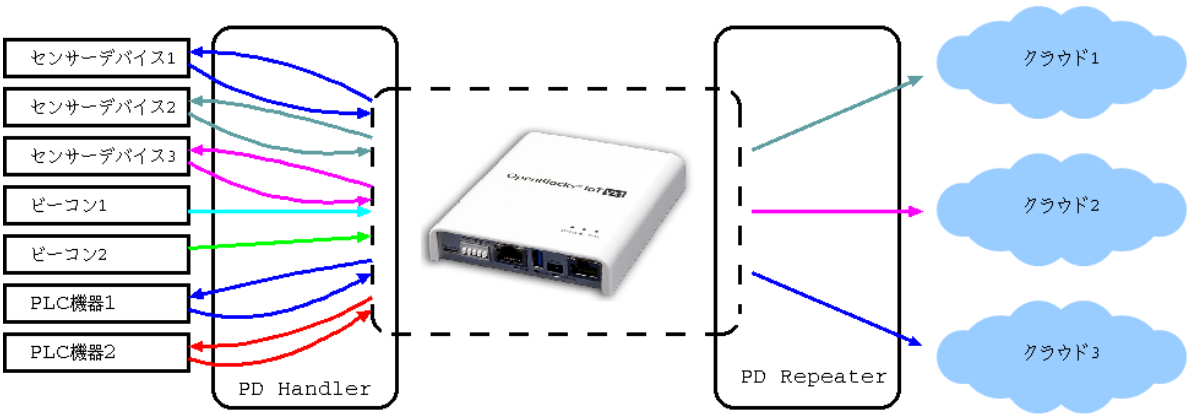
本設定には、WEB ブラウザが使用可能なクライアント装置 (PC やスマートフォン、タブレット等) が必要になります。

また、WEB ユーザーインターフェース (以下、WEB UI) 自体については *WEB UI* **セットアップガイド** を参照してください。

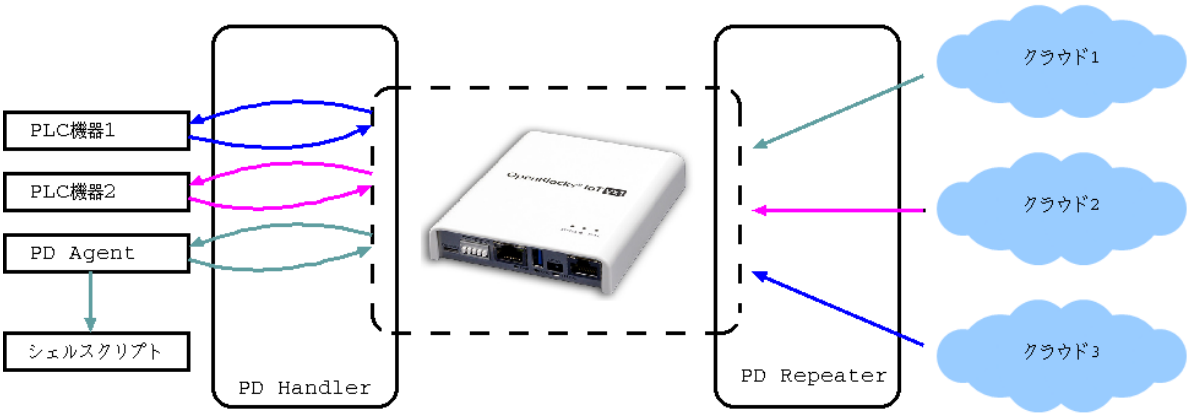
第16章　IoTデータ制御機能

OpenBlocks シリーズの IoT データ制御機能は、BLE・UART 等のセンサーデバイス並びに Modbus 対応 PLC 機器からデータを取得しクラウド等へ情報を送信する収集機能と、クラウド等からメッセージを受信し Modbus 対応 PLC 機器等を制御する下流方向の制御機能をサポートしています。センサーデバイス等のサポート状況については、 弊社 WEB ページ を参照してください。

収集機能は各デバイス等からデータを取得し、各送信先のクラウド等へ情報を送信します。データを一時バッファーとして OpenBlock シリーズ内に保存している為、ネットワーク障害等が発生しても、再送信が行える為データを安全に送信することが出来ます。



下流方向制御機能はクラウドから制御メッセージを受け取り、Modbus 対応 PLC 等を制御することが可能です。



IoT データ制御機能を使用するためには、サービス機能の基本設定と IoT データ制御パッケージのインストールが必要となります。

第17章　サービス機能の基本設定

IoT データ制御機能を使用するためには、WEB UI の「サービス」→「基本」タブより、サービス機能の基本設定として BT インターフェースの制御及び各種デバイスの登録等を行う必要があります。

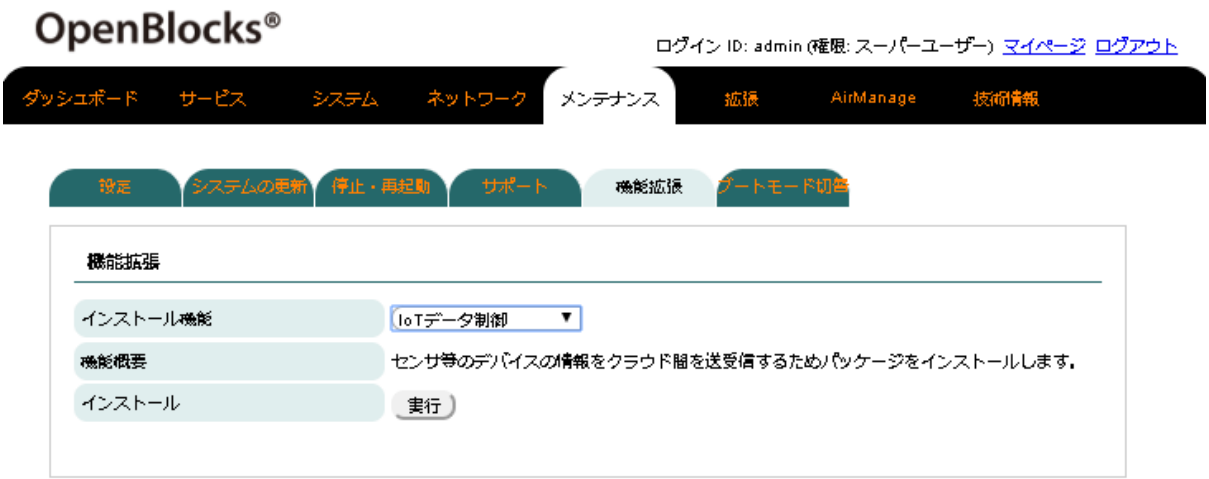
サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

第18章　IoTデータ制御パッケージのインストール

IoT データ制御機能を使用するためには、WEB UI の「メンテナンス」→「機能拡張」タブより、IoT データ制御パッケージをインストールする必要があります。

注釈:

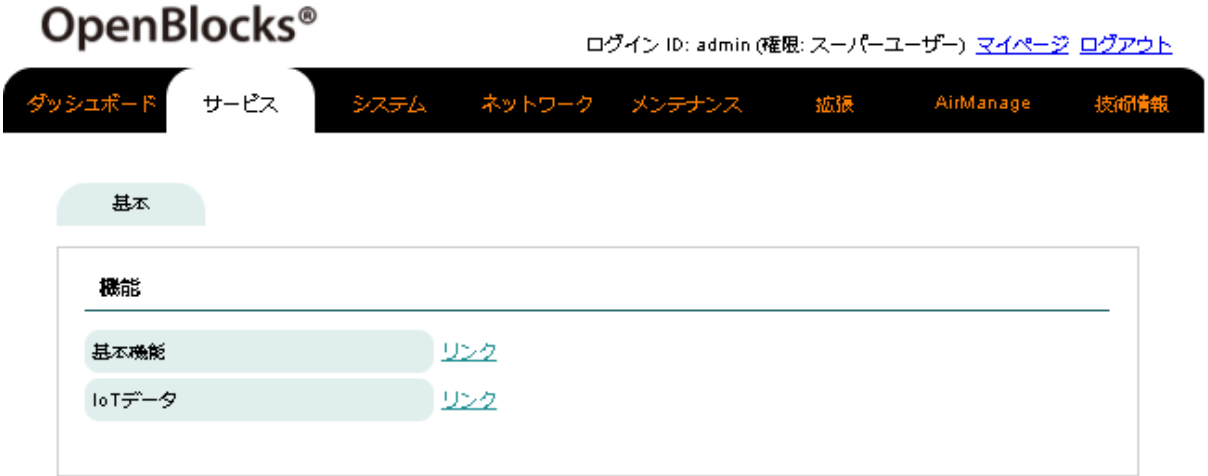
- IoT データ制御は Node-RED との連携が可能です。そのため、Node-RED との連携をする場合には、Node-RED についてもインストールを行ってください。
- IoT データ制御は Azure IoT Edge に対しても送信が可能です。同一の OpenBlocks 内の Azure IoT Edge に対してデータを送信する場合には、Azure IoT Edge についてもインストールを行ってください。



メンテナンス」→「機能拡張」タブの「インストール機能」で「IoT データ制御」を選択し、「実行」をクリックします。IoT データ制御パッケージが既にインストールされている場合は、「インストール機能」の選択肢として「IoT データ制御」が表示されません。

第19章　IoTデータ制御のアプリ設定

IoT データ制御パッケージがインストールされていると WEB UI の「サービス」→「基本」タブに「IoT データ」が表示されます。



「サービス」→「基本」タブより「IoT データ」リンクをクリックすると、ルートタブが IoT データ制御の設定を行う「ダッシュボード」／「基本」／「IoT データ」に切り替わります。

注釈:

- 「サービス」→「基本」タブに表示されるリンクは、「メンテナンス」→「機能拡張」タブよりインストールされたパッケージにより異なります。

19.1 アプリ起動制御

WEB UI の「IoT データ」→「アプリ設定」タブより、使用するアプリケーションの起動制御とコンフィグ設定モードの設定を行います。

「デフォルト使用アプリモード表示」をチェックすると、「デフォルトアプリ起動制御」の設定メニューを表示します。
各アプリケーションの起動制御を「使用する」に設定するためには、デフォルト側の設定が「使用する」に設定されている必要があります。

図 1: 「デフォルト使用アプリモード表示」チック時の表示例

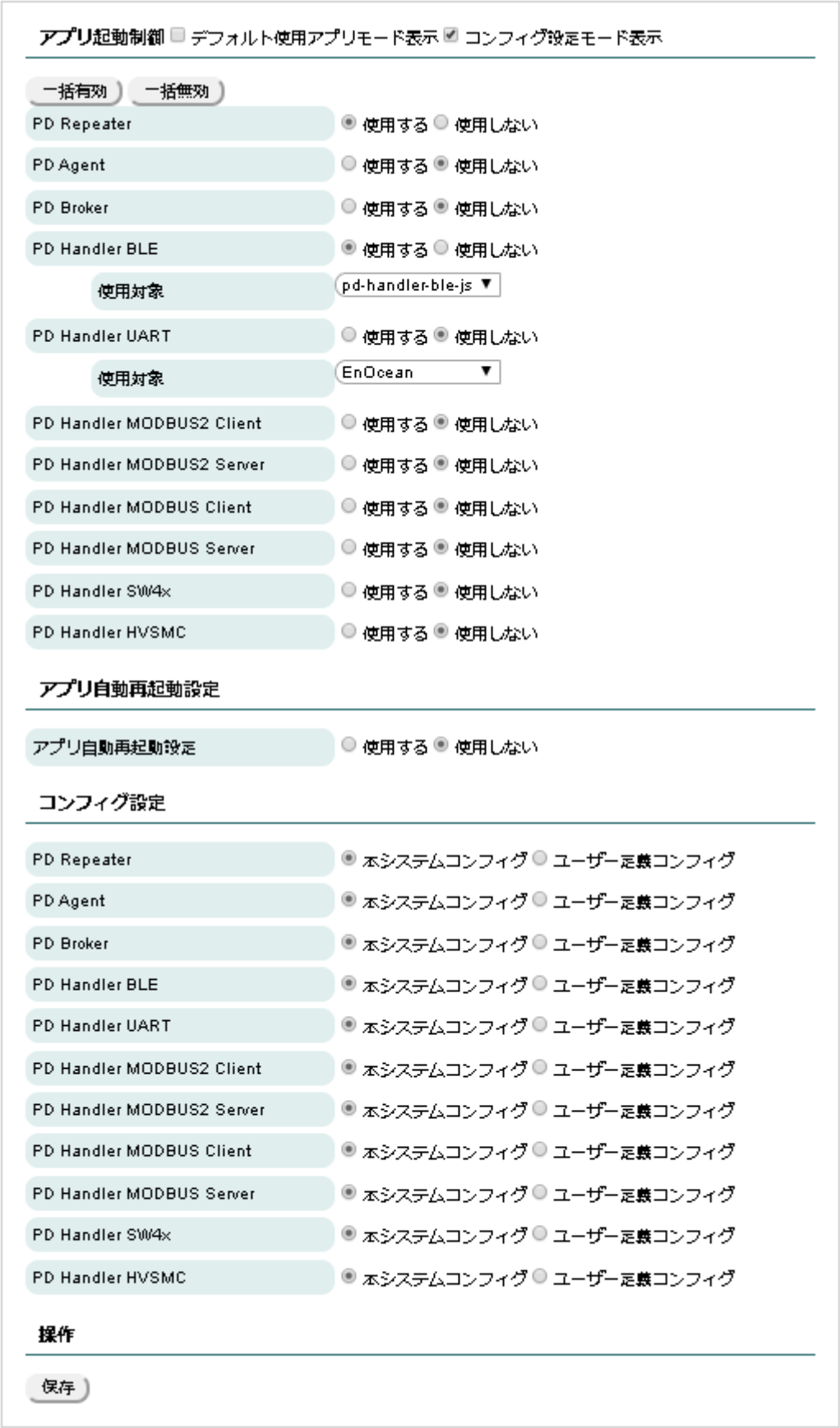


図 2: 「コンフィグ設定モード表示」チック時の表示例

「コンフィグ設定モード表示」をチェックすると本 WEB UI により設定されるコンフィグを使用するか、ユーザー定義のコンフィグを使用するか切り替える「コンフィグ設定」メニューを表示します。

「一括有効」「一括無効」は、全ての設定を一括して「使用する」もしくは「使用しない」に設定します。

表 1: アプリ起動制御の設定項目

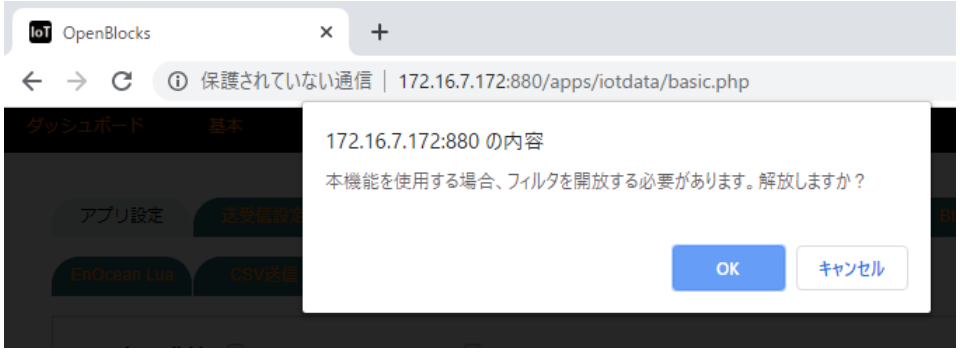
設定項目	説明
PD Repeater	クラウドに対しデータもしくは制御メッセージを送受するアプリケーションです。
PD Agent	PD Reperter を介しクラウドから制御メッセージを受け設定されたシェルスクリプトもしくは実行オブジェクトを実行するアプリケーションです。
PD Broker	データもしくは制御メッセージを複数のアプリケーションに分配するアプリケーションです。
PD Handler BLE	BT デバイスセンサーもしくはビーコンからデータを受け取るアプリケーションです。「使用対象」として C 言語版と Java スクリプト版のいずれかを選択できます。C 言語版の方がハイパフォーマンスですがビーコン及び非コネクション型 BLE センサーのみの対応となります。C 言語版で対応していない非コネクション型 BLE センサーについては、*Lua* **拡張** 機能で対応することが可能です。
PD Handler UART	URAT 接続のデバイスからデータを受け取るアプリケーションです。「使用対象」として次のモジュールを選択することができます。 • EnOcean • Wi-SUN (B ルート)
PD Handler MODBUS2 Client	モデリングに基づき MODBUS プロトコルを用いて PLC 機器のレジスタを読み書きするアプリケーションです。 *1
PD Handler MODBUS2 Server	モデリングに基づき PLC 機器から MODBUS プロトコルによる接続を待ち受けるアプリケーションです。 *1
PD Handler MODBUS Client	MODBUS プロトコルを用いて PLC 機器のレジスタを読み書きするアプリケーションです。 *1
PD Handler MODBUS Server	PLC 機器から MODBUS プロトコルによる接続を待ち受けるアプリケーションです。 *1
PD Handler SW4x	SII 社製ミスター省エネ無線ネットワークに接続し無線センサーのデータを取得する。アプリケーションです。
PD Handler HVSMC	高圧スマート電力量メータ (B ルート) から ECHONET Lite と AIF の仕様に基きデータを取得するアプリケーションです。 *2

注釈:

1. PD Handler MODBUS の 1 系と 2 系のアプリケーションを、同時に使用することはできません。
2. PD Handler HVSMC のラジオボタンは ECHONET Lite の認定を受けた同アプリケーションが使用可能な機種 (Open Blocks IoT EX1,VX2) でのみ表示されます。

■ IP フィルタの開放について

PD Handler HVSMC 機能を「使用する」に設定し保存した場合、フィルタ開放確認のウィンドウが表示されます。　ウィンドウ画面にて適用する旨を選択した場合、自動でフィルタが恒久的に反映されます。



PD Handler MODBUS Server または PD Handler MODBUS2 Server 機能を「使用する」に設定し保存した場合、上記のフィルタ開放確認のウィンドウに続き、フィルタ開放ポリシー設定 [Modbus] のウィンドが表示されます。

「ローカルエリアネットワークのみ」もしくは「全て」のいずれかを選択して下さい。

一旦、開放されたフィルタは、これらの機能を「使用しない」に設定し保存しても開放されたままとなります。

開放されたフィルタを閉じたい場合、もしくはフィルタ開放ポリシーを変更したい場合は、WEB UI の「システム」→「フィルター」タブの「フィルター開放設定」メニューから設定することができます。

19.2 アプリ自動再起動設定

「アプリ自動再起動設定」は、アプリケーションを定期的に自動再起動する機能です。
ユーザーが独自に開発したアプリケーション等で使用メモリー量が徐々に増加してしまう場合等に用います。

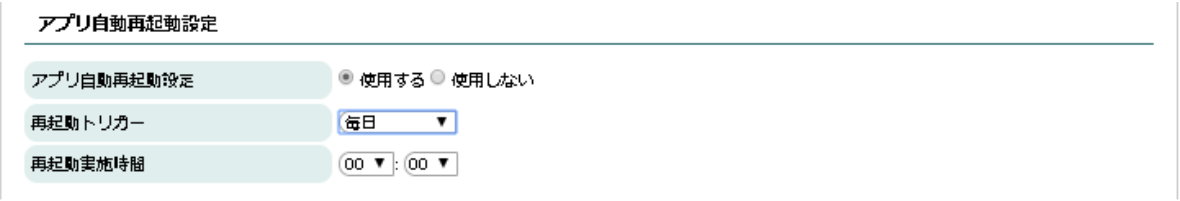


図 3: 「アプリ自動再起動設定」「再起動トリガー」を「毎日」とした場合の表示例

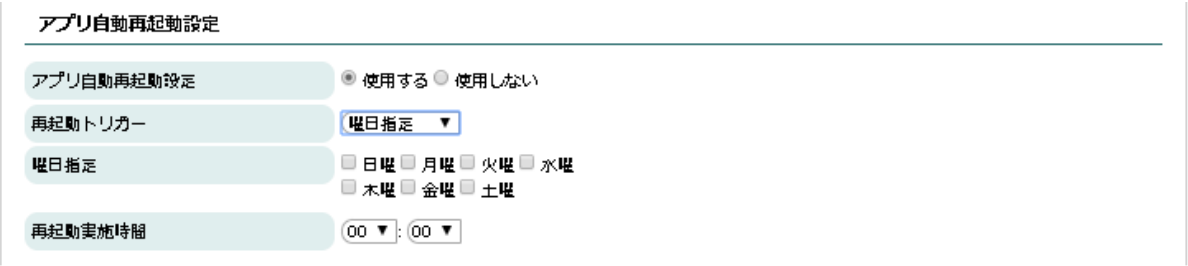


図 4: 「アプリ自動再起動設定」「再起動トリガー」を「曜日指定」とした場合の表示例

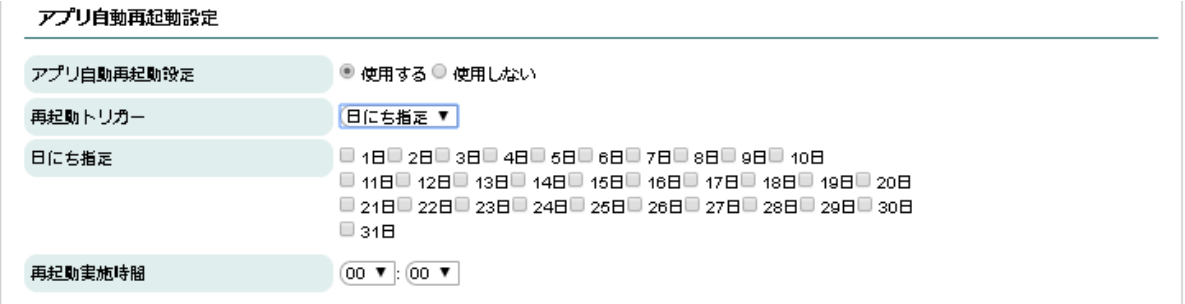


図 5: 「アプリ自動再起動設定」「再起動トリガー」を「日にち指定」とした場合の表示例

表 2: アプリ自動再起動設定の設定項目

設定項目	説明
アプリ自動再起動設定	「使用する」もしくは「使用しない」に設定します。
再起動トリガー	「毎日」、「曜日指定」、「日にち指定」のいずれかを選択します。
曜日指定	再起動させる曜日を指定します。複数日指定することができます。
日にち指定	再起動させる日にちを指定します。複数日指定することができます。
再起動実施時間	再起動させる時刻を指定します。

第20章　送受信設定（PD Repeater）

WEB UI の「IoT データ」→「送受信設定」タブより、データの送信先もしくは制御メッセージの受信先の設定を行います。
送受信設定には、送受信先毎に「送受信設定」メニューより設定される個々のデバイスに依存しない設定と各デバイス設定メニューより設定されるデバイス毎の設定があります。

20.1 送受信設定

「IoT データ」→「送受信設定」タブの「送受信設定」メニューより、送受信先の選択と送受信設定の内個々のデバイスに依存しない設定を行います。
本体内 (lo) を除き最大４つの送受信先を選択することができます。

表 1: 送受信設定の設定項目

設定項目	説明
本体内 *(lo)*	BLE デバイスの JSON テーブルと温度／湿度グラフを「IoT データ」→「データ表示」タブに出力します。
DEXPF(dexpf)	送受信先として DEXPF を選択します。
DEXPF[Websocket](dexpf_ws)	送受信先として DEXPF を選択します。接続プロトコルとして Websocket over MQTT を用います。
MS Azure IoT Hub(iothub)	送受信先として Microsoft Azure IoT Hub を選択します。尚、Azure IoT Edge に対して送信する場合には、本項目を選択します。
MS Azure IoT Hub[Websocket](iothub_ws)	送受信先として Microsoft Azure IoT Hub を選択します。接続プロトコルとして Websocket over MQTT を用います。
AWS IoT(awsiot)	送受信先として Amazon AWS IoT を選択します。
AWS IoT[Websocket](awsiot_ws)	送受信先として Amazon AWS IoT を選択します。接続プロトコルとして Websocket over MQTT を用います。
Google IoT Core(iotcore)	送受信先として Google IoT Core を選択します。
Watson IoT for Gateway(w4g)	送受信先として IBM Watson IoT for Gateway を選択します。
MS Azure Event hubs(eventhub)	送信先として Microsoft Azure Event hubs を選択します。
Amazon Kinesis(kinesis)	送信先として Amazon Kinesis を選択します。
Watson IoT for Device(w4d)	送受信先として IBM Watson IoT for Device を選択します。
Toami for DOCOMO(t4d)	送信先として NTT docomo Toami for DOCOMO を選択します。
SoftBank **スマ可視専用クラウド** *(sbiot)*	送信先として SoftBank スマ可視専用クラウドを選択します。
KDDI IoT **クラウド** *Standard(kddi_std)*	送信先として KDDI IoT クラウド Standard を選択します。
PH **社独自仕様** *WEB* **サーバー** *(pd_web)*	送受先として弊社独自仕様の WEB サーバーを選択します。
WEB **サーバー** *(web)*	送信先として汎用の WEB サーバーを選択します。
MQTT **サーバー** *(mqtt)*	送受信先として汎用の MQTT サーバーを選択します。
TCP(ltcp)	送受信先として汎用の TCP サーバーを選択します。
PD Exchange(pd_ex)	送受信先として PD Exchange を選択します。
ドメインソケット *(lsocket)*	送信先として UNIX ドメインソケットを選択します。Node-RED パッケージがインストールされている場合のみ表示されます。

20.1.1 送受信先毎の設定

送受信先毎の「送受信設定」メニューにおける設定と各デバイス設定メニューにおける設定を説明します。

本体内 (lo)

BLE デバイスの JSON テーブルと温度／湿度グラフを「IoT データ」→「データ表示」タブに出力します。本体内 (lo) は、BLE デバイスのみに適用されます。

■ 送受信設定メニューにおける設定

表 2: 送受信設定メニューにおける設定項目

設定項目	説明
	「使用する」／「使用しない」以外の設定はありません。
デバイス一括設定	BLE デバイス情報送信設定において送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ BLE デバイス情報送信設定メニューにおける設定

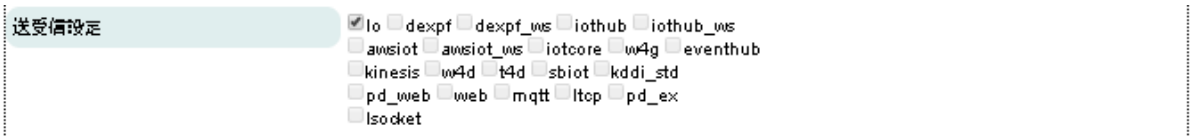


表 3: BLE デバイス情報送信設定メニューにおける設定項目

設定項目	説明
	「送受信設定」の「lo」をチェックする以外の設定項目はありません。

DEXPF(dexpf)

■ 送受信設定メニューにおける設定

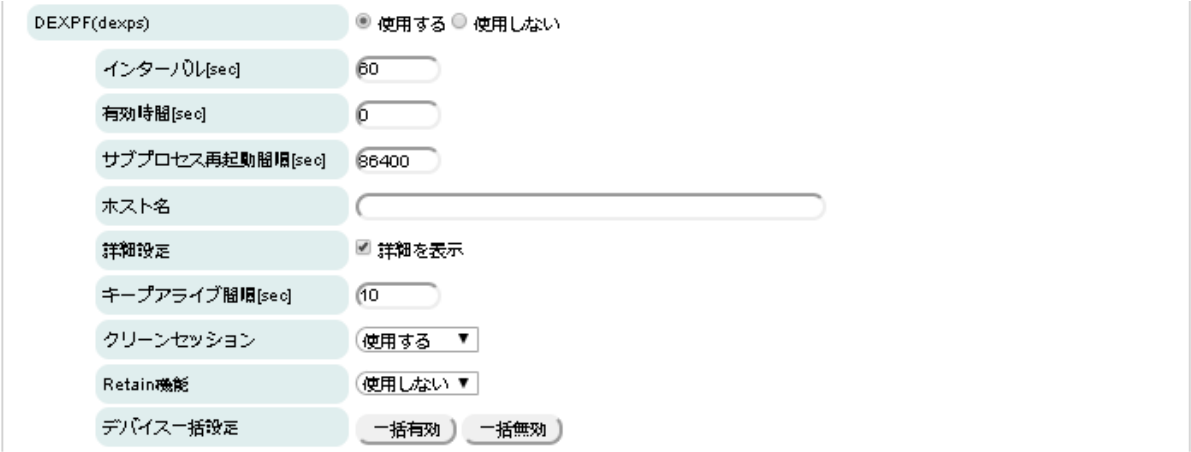


表 4: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ホスト名	DEXPF のホスト名を設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
クリーンセッション	MQTT プロトコルのクリーンセッションの使用／不使用を選択します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

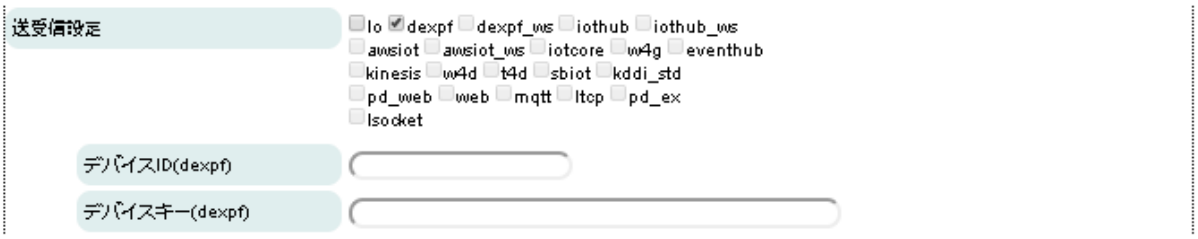


表 5: 各デバイス設定メニューにおける設定項目

設定項目	説明
デバイス ID(dexpf)	DEXPF のデバイス ID を設定します。
デバイスキー (dexpf)	DEXPF のデバイスキーを設定します。

DEXPF[Websocket](dexpf_ws)

■ 送受信設定メニューにおける設定

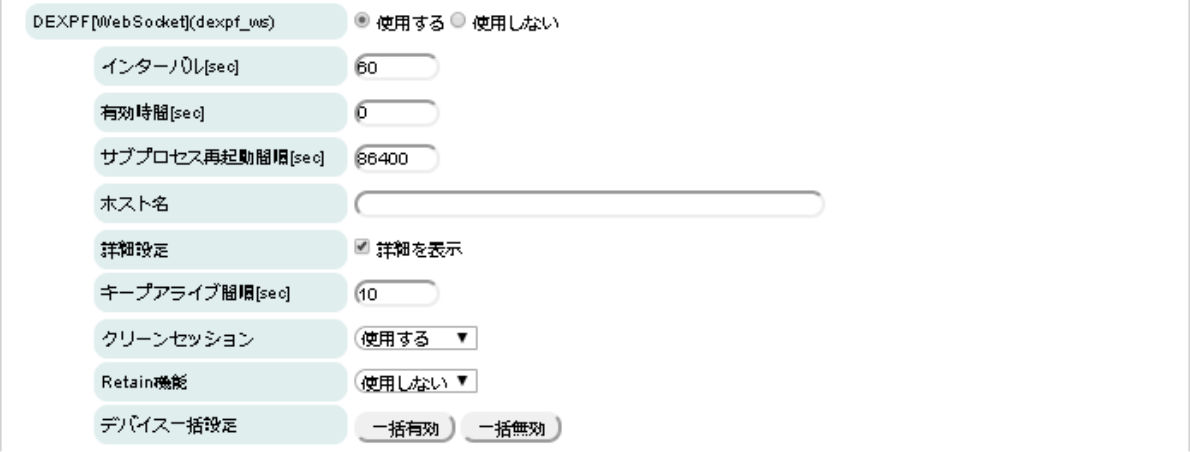


表 6: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ホスト名	DEXPF のホスト名を設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
クリーンセッション	MQTT プロトコルのクリーンセッションの使用／不使用を選択します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

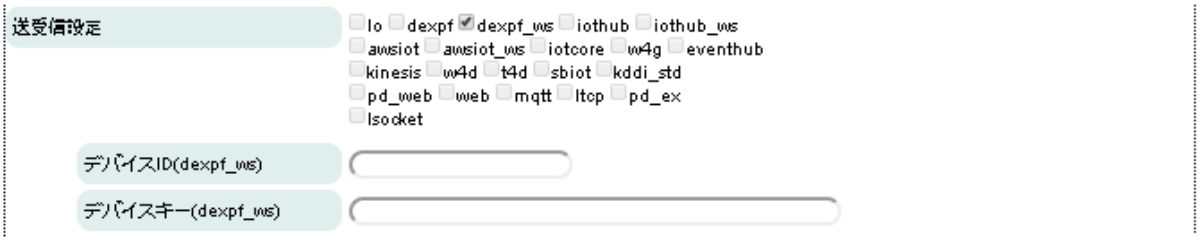


表 7: 各デバイス設定メニューにおける設定項目

設定項目	説明
デバイス ID(dexpf_ws)	DEXPF のデバイス ID を設定します。
デバイスキー (dexpf_ws)	DEXPF のデバイスキーを設定します。

MS Azure IoT Hub(iothub)

■ 送受信設定メニューにおける設定

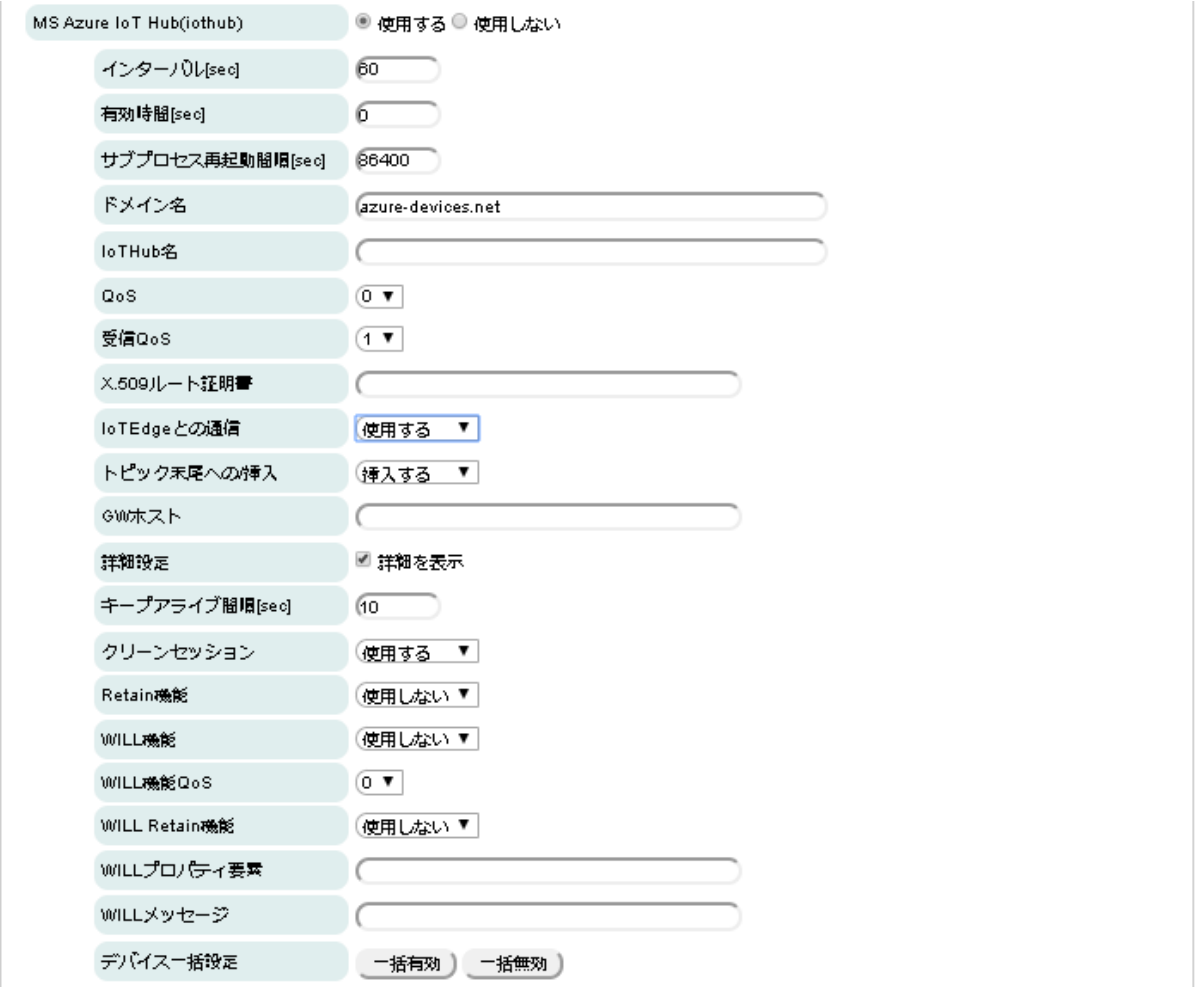


表 8: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ドメイン名	接続先の IoT Hub のドメイン名を設定します。
IoT Hub 名	接続先の IoT Hub 名を設定します。
QoS	IoT Hub へデータを送信する際の MQTT プロトコルの QoS を設定します。
受信 QoS	IoT Hub から制御メッセージを受信する際の MQTT プロトコルの QoS を設定します。
X.509 ルート証明書	X.509 認証を用いる場合のルート証明書ファイルのパス名を設定します。
IoT Edge との通信	IoT Edge の使用／不使用を選択します。
トピック末尾への/挿入	IoT Edge へデータを送信する際の MQTT トピックの末尾に'/' を挿入するか選択します。(IoT Hub との通信時には影響はありません) *1
GW ホスト	IoT Edge のゲートウェイホスト名を設定します。（IoT Edge 使用時のみ）尚、別ホストにて稼働している IoT Edge データ送信する場合、IP アドレスまたは FQDN を指定してください。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
クリーンセッション	MQTT プロトコルのクリーンセッションの使用／不使用を選択します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
WILL 機能	MQTT プロトコルの LWT(LastWill and Testament) 機能の使用／不使用を選択します。
WILL QoS 機能	MQTT プロトコルが WILL メッセージを送出する際の QoS を設定します。
WILL Retain 機能	MQTT プロトコルの WILL Retain 機能の使用／不使用を選択します。
WELL プロパティ要素	MQTT プロトコルの WILL トピックに用いられる Azure IoT Hub のプロパティ・バッグを設定します。
WLL メッセージ	MQTT プロトコルの WILL メッセージを設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

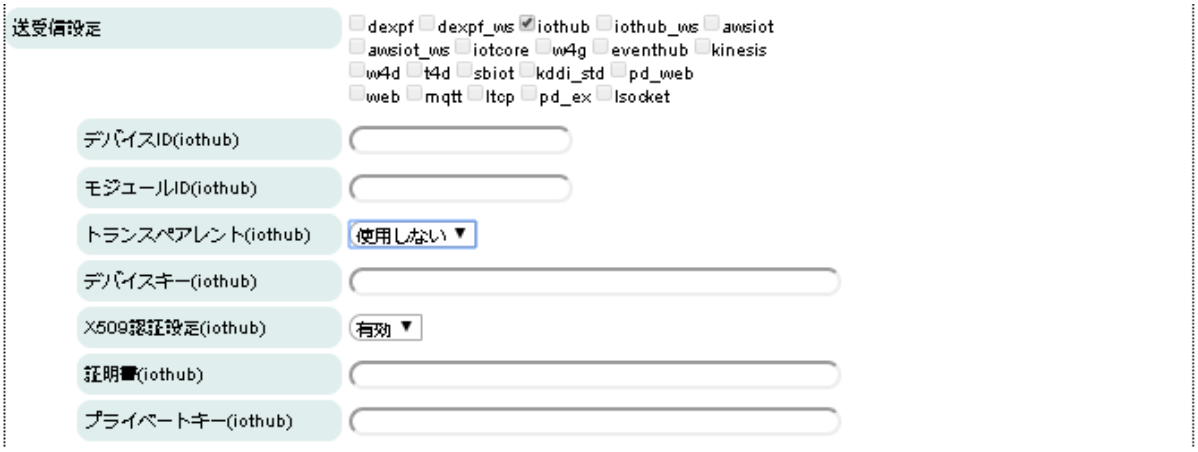


表 9: 各デバイス設定メニューにおける設定項目

設定項目	説明
デバイス ID(iothub)	IoT Hub のデバイス ID を設定します。
モジュール ID(iothub)	IoT Edge を使用する場合のモジュール ID を設定します。
トランスペアレント (iothub)	IoT Edge の透過モードを用いるか否かを選択します。 *2
デバイスキー (iothub)	IoT Hub のデバイスキーを設定します。
X.509 認証設定 (iothub)	X.509 認証を使用する場合は ”有効 ”を選択します。
証明書 (iothub)	X.509 認証のための証明書ファイルのパス名を設定します。
プライベートキー (iothub)	X.509 認証のためのプライベートキーのパス名を設定します。

注釈:

1. 2018/09/14 現在、トピック末尾に'/' ある場合、IoT Edge 経由 IoT Hub へのデータ送信に失敗します。そのため、トピック末尾'/' に挿入しないでください。
2. IoT Edge のトランスペアレント（透過モード）を用いる場合、モジュール ID は無視されます。

- IoT Edge を使用しない場合のデバイス ID と使用する場合のデバイス ID は、Azure ポータルサイト上での登録方法が異なります。前者は IoT Hub のデバイスエクスプローラから、後者は IoT Hub の IoT Edge から登録します。
- IoT Edge を使用する場合のゲートウェイホスト名 (GW ホスト) とモジュール ID は、WEB UI の「サービス」→「Azure IoT Edge」→「Edge 環境変数」タブにて表示される IoT Edge の環境変数に与えられている値を用います。
- OpenBlocks シリーズ BX シリーズと OpenBlocks シリーズ EX シリーズについては、Docker もしくは Mody が動作しないため、IoT Edge の GW ホストとしてローカルホストを使うことは出来ません。
- X.509 ルート証明書・証明書・プライベートキーは WEB UI の「システム」→「ファイル管理」タブにてアップロードしてください。

MS Azure IoT Hub[Websocket](iothub_ws)

■ 送受信設定メニューにおける設定

表 10: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ドメイン名	接続先の IoT Hub のドメイン名を設定します。
IoT Hub 名	接続先の IoT Hub 名を設定します。
QoS	IoT Hub へデータを送信する際の MQTT プロトコルの QoS を設定します。
受信 QoS	IoT Hub から制御メッセージを受信する際の MQTT プロトコルの QoS を設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
クリーンセッション	MQTT プロトコルのクリーンセッションの使用／不使用を選択します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
WILL 機能	MQTT プロトコルの LWT(LastWill and Testament) 機能の使用／不使用を選択します。
WILL QoS 機能	MQTT プロトコルが WILL メッセージを送出する際の QoS を設定します。
WILL Retain 機能	MQTT プロトコルの WILL Retain 機能の使用／不使用を選択します。
WELL プロパティ要素	MQTT プロトコルの WILL トピックに用いられる Azure IoT Hub のプロパティ・バッグを設定します。
WLL メッセージ	MQTT プロトコルの WILL メッセージを設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

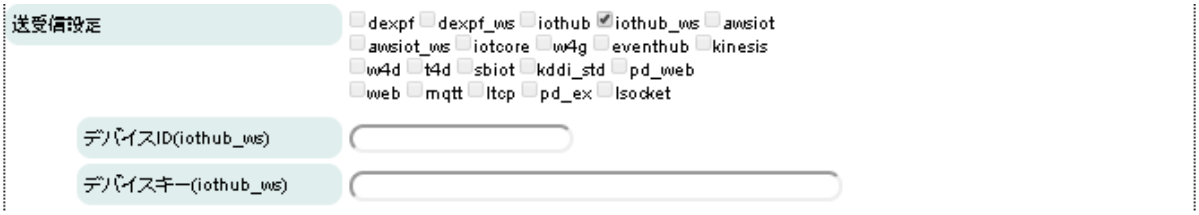


表 11: 各デバイス設定メニューにおける設定項目

設定項目	説明
デバイス ID(iothub_ws)	IoT Hub のデバイス ID を設定します。
デバイスキー (iothub_ws)	IoT Hub のデバイスキーを設定します。

注釈:

- IoT Edge には対応しておりません。

AWS IoT(awsiot)

■ 送受信設定メニューにおける設定

表 12: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
送信先ホスト	接続先の AWS IoT のホスト名を設定します。
ポート番号	接続先の AWS IoT のポート番号を設定します。
QoS	AWS IoT へデータを送信する際の MQTT プロトコルの QoS を設定します。
受信 QoS	AWS IoT から制御メッセージを受信する際の MQTT プロトコルの QoS を設定します。
ルート証明書	AWS IoT へ接続する際のルート証明書ファイルのパス名を設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
WILL 機能	MQTT プロトコルの LWT(LastWill and Testament) 機能の使用／不使用を選択します。
WILL QoS 機能	MQTT プロトコルが WILL メッセージを送出する際の QoS を設定します。
WILL Retain 機能	MQTT プロトコルの WILL Retain 機能の使用／不使用を選択します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

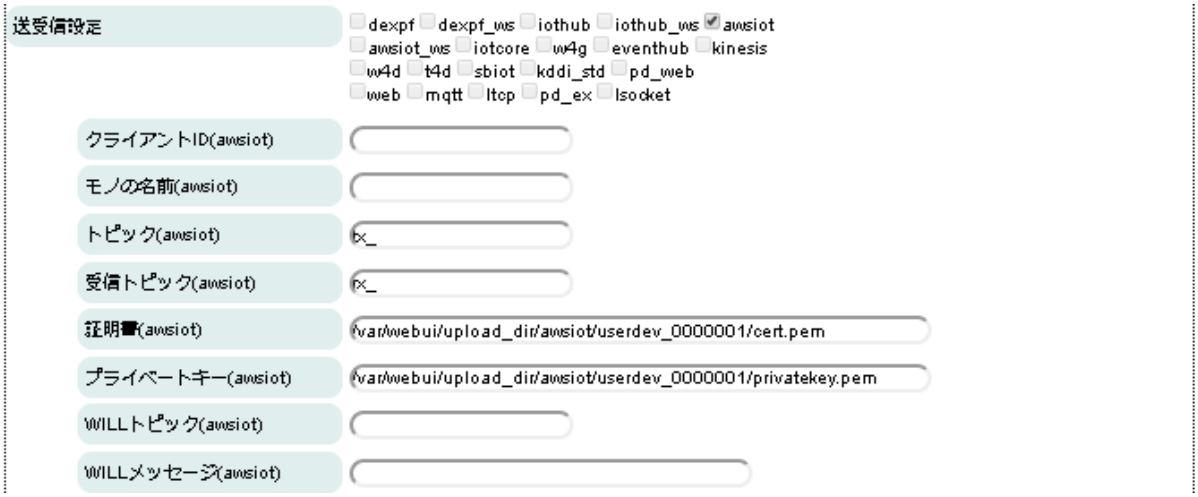


表 13: 各デバイス設定メニューにおける設定項目

設定項目	説明
クライアント ID(awsiot)	AWS IoT のクライアント ID を設定します。
モノの名前 (awsiot)	AWS IoT の Device Shadows 機能に使用する thingName を指定します。Device Shadows 機能の利用については第 2 − 5 − 5 − 2 章を参照して下さい。
トピック (awsiot)	AWS IoT へデータを送る際に用いる MQTT プロトコルのトピックを設定します。
受信トピック (awsiot)	AWS IoT から制御メッセージを待ち受けるために用いる MQTT プロトコルのトピックを設定します。
証明書 (awsiot)	AWS IoT へ接続する際の証明書ファイルのパス名を設定します。
プライベートキー (awsiot)	AWS IoT へ接続する際のプライベートキーファイルのパス名を設定します。
WELL トピック (awsiot)	MQTT プロトコルの WILL トピックを設定します。
WLL メッセージ (awsiot)	MQTT プロトコルの WILL メッセージを設定します。

注釈:

- ルート証明書・証明書・プライベートキーは WEB UI の「システム」→「ファイル管理」タブにてアップロードしてください。

AWS IoT[Websocket](awsiot_ws)

■ 送受信設定メニューにおける設定

表 14: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
送信先ホスト	接続先の AWS IoT のホスト名を設定します。
ポート番号	接続先の AWS IoT のポート番号を設定します。
QoS	AWS IoT へデータを送信する際の MQTT プロトコルの QoS を設定します。
受信 QoS	AWS IoT から制御メッセージを受信する際の MQTT プロトコルの QoS を設定します。
アクセス ID	AWS IoT のアクセス ID を設定します。
アクセスキー	AWS IoT のアクセスキーを設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
WILL 機能	MQTT プロトコルの LWT(LastWill and Testament) 機能の使用／不使用を選択します。
WILL QoS 機能	MQTT プロトコルが WILL メッセージを送出する際の QoS を設定します。
WILL Retain 機能	MQTT プロトコルの WILL Retain 機能の使用／不使用を選択します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

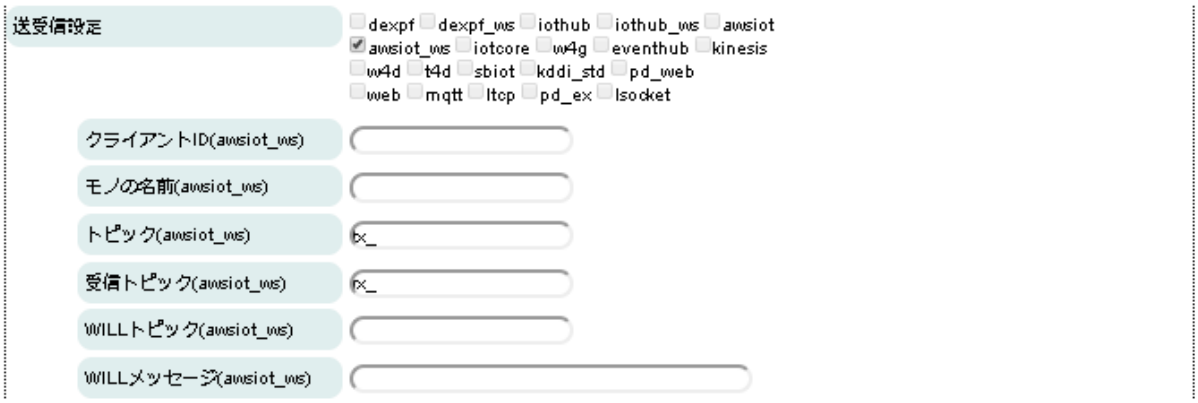


表 15: 各デバイス設定メニューにおける設定項目

設定項目	説明
クライアント ID(awsiot_ws)	AWS IoT のクライアント ID を設定します。
モノの名前 (awsiot_ws)	AWS IoT の Device Shadows 機能に使用する thingName を指定します。Device Shadows 機能の利用については第 2 − 5 − 5 − 2 章を参照して下さい。
トピック (awsiot_ws)	AWS IoT へデータを送る際に用いる MQTT プロトコルのトピックを設定します。
受信トピック (awsiot_ws)	AWS IoT から制御メッセージを待ち受けるために用いる MQTT プロトコルのトピックを設定します。
WELL トピック (awsiot_ws)	MQTT プロトコルの WILL トピックを設定します。
WLL メッセージ (awsiot_ws)	MQTT プロトコルの WILL メッセージを設定します。

Google IoT Core(iotcore)

■ 送受信設定メニューにおける設定

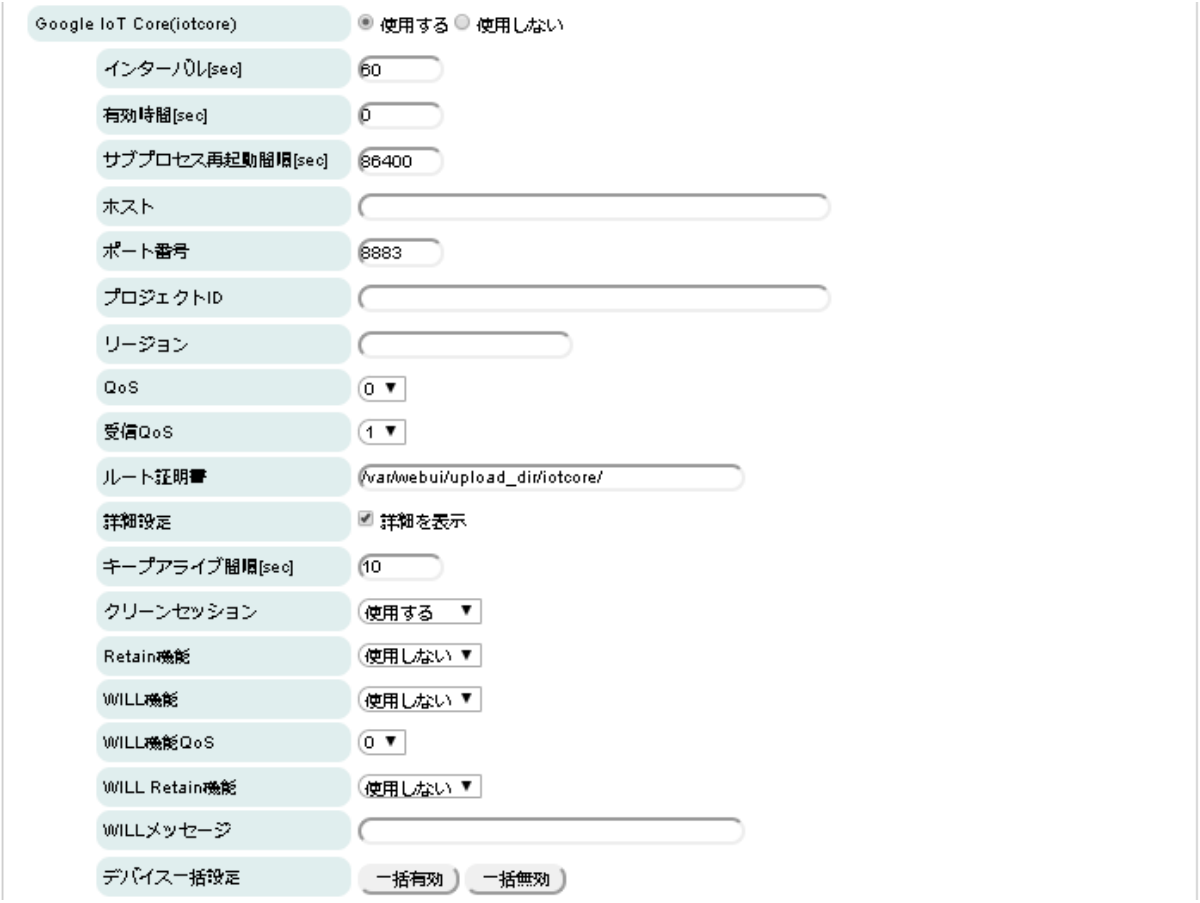


表 16: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ホスト	接続先の Google IoT Core のホスト名を設定します。
ポート番号	接続先の Google IoT Core のポート番号を設定します。
プロジェクト ID	Google IoT Core のプロジェクト ID を設定します。
リージョン	Google IoT Core のリージョンコードを設定します。
QoS	Google IoT Core へデータを送信する際の MQTT プロトコルの QoS を設定します。
受信 QoS	Google IoT Core から制御メッセージを受信する際の MQTT プロトコルの QoS を設定します。
ルート証明書	Google IoT Core へ接続する際のルート証明書ファイルのパス名を設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
クリーンセッション	MQTT プロトコルのクリーンセッションの使用／不使用を選択します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
WILL 機能	MQTT プロトコルの LWT(LastWill and Testament) 機能の使用／不使用を選択します。
WILL QoS 機能	MQTT プロトコルが WILL メッセージを送出する際の QoS を設定します。
WILL Retain 機能	MQTT プロトコルの WILL Retain 機能の使用／不使用を選択します。
WLL メッセージ	MQTT プロトコルの WILL メッセージを設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

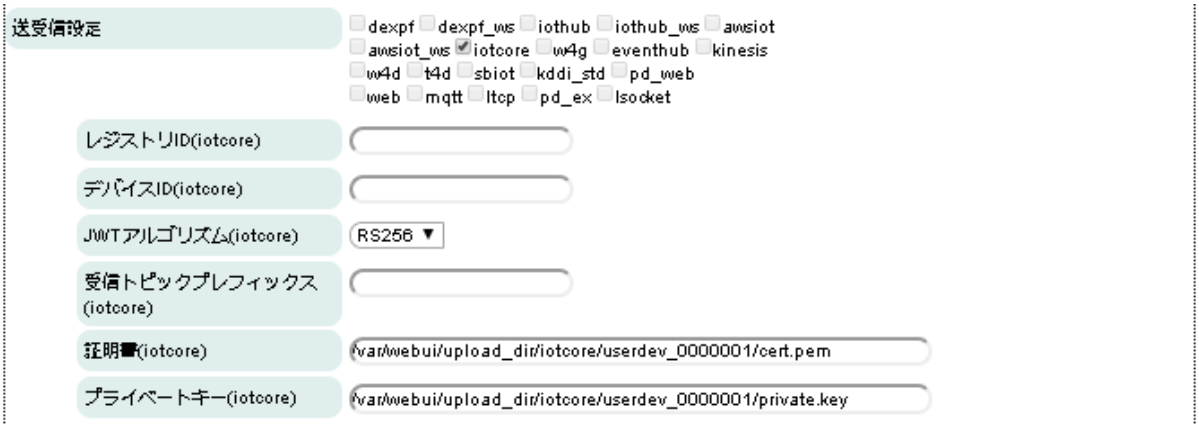


表 17: 各デバイス設定メニューにおける設定項目

設定項目	説明
レジストリ ID(iotcore)	Google IoT Core のクライアント ID を設定します。
デバイス ID(iotcore)	Google IoT Core のデバイス ID を設定します。
JWT アルゴリズム (iotcore)	Google IoT Core への接続に用いる JWT アルゴリズムを選択します。
証明書 (iotcore)	Google IoT Core へ接続する際の証明書ファイルのパス名を設定します。
プライベートキー (iotcore)	Google IoT Core へ接続する際のプライベートキーファイルのパス名を設定します。

注釈:

- ルート証明書・証明書・プライベートキーは WEB UI の「システム」→「ファイル管理」タブにてアップロードしてください。

Watson IoT for Gateway(w4g)

■ 送受信設定メニューにおける設定

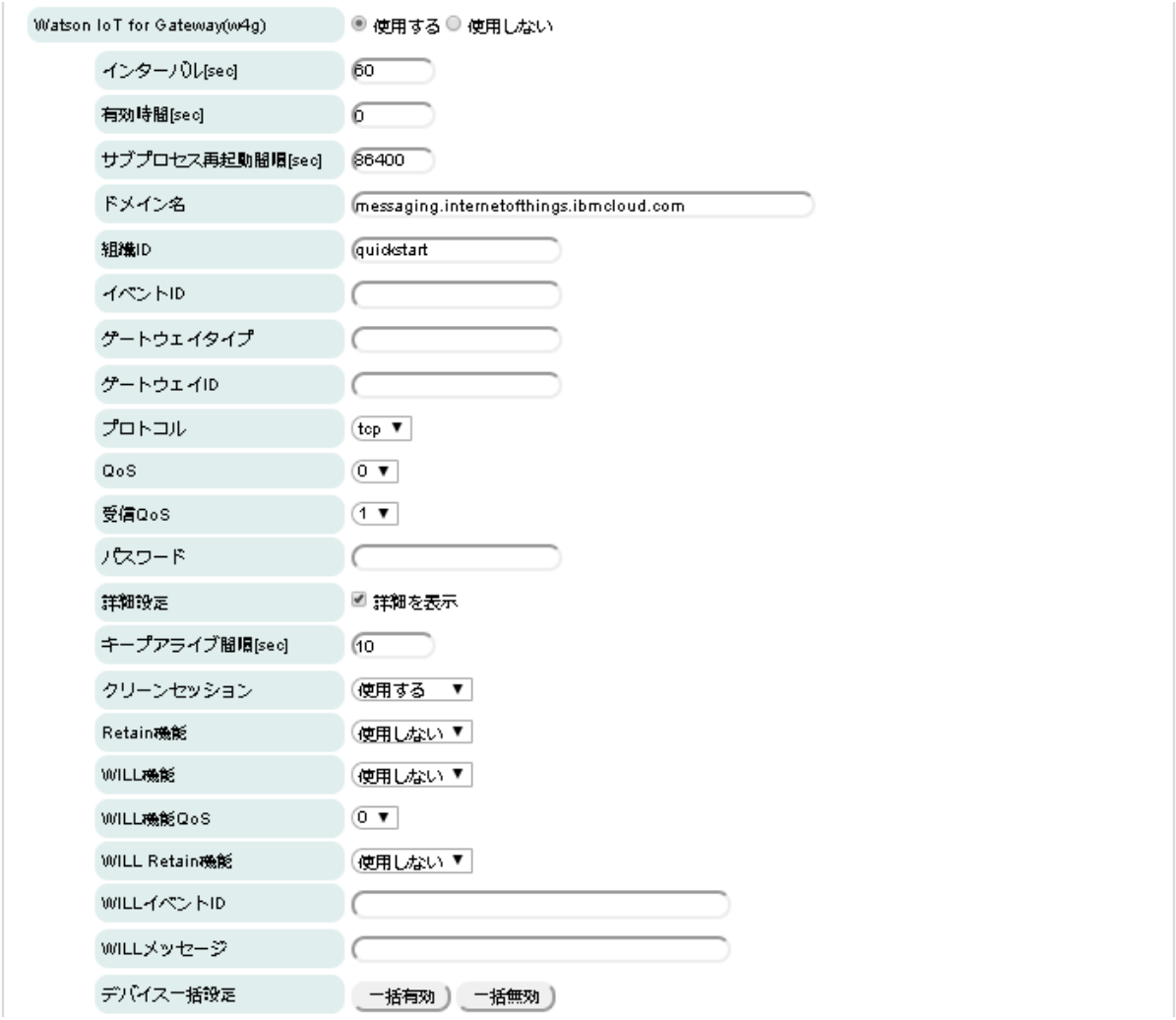


表 18: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ドメイン名	接続先の Watson IoT for Gateway のドメイン名を設定します。
組織 ID	Watson IoT for Gateway の組織 ID を設定します。
イベント ID	Watson IoT for Gateway のイベント ID を設定します。
ゲートウェイタイプ	Watson IoT for Gateway のゲートウェイタイプを設定します。
ゲートウェイ ID	Watson IoT for Gateway のゲートウェイ ID を設定します。
プロトコル	接続に用いるプロトコル (TCP/SSL) を選択します。
QoS	Watson IoT for Gateway へデータを送信する際の MQTT プロトコルの QoS を設定します。
受信 QoS	Watson IoT for Gateway から制御メッセージを受信する際の MQTT プロトコルの QoS を設定します。
パスワード	Watson IoT for Gateway への接続に用いるパスワードを設定します。
トラストストア	Watson IoT for Gateway へ接続する際のルート証明書ファイルのパス名を設定します。
キーストア	Watson IoT for Gateway へ接続する際のクライアント証明書ファイルのパス名を設定します。
プライベートキー	Watson IoT for Gateway へ接続する際のプライベートキーファイルのパス名を設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
クリーンセッション	MQTT プロトコルのクリーンセッションの使用／不使用を選択します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
WILL 機能	MQTT プロトコルの LWT(LastWill and Testament) 機能の使用／不使用を選択します。
WILL QoS 機能	MQTT プロトコルが WILL メッセージを送出する際の QoS を設定します。
WILL Retain 機能	MQTT プロトコルの WILL Retain 機能の使用／不使用を選択します。
WELL イベント ID	MQTT の WILL トピックに用いられる Watson IoT for Gateway のイベント ID を設定します。
WLL メッセージ	MQTT プロトコルの WILL メッセージを設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

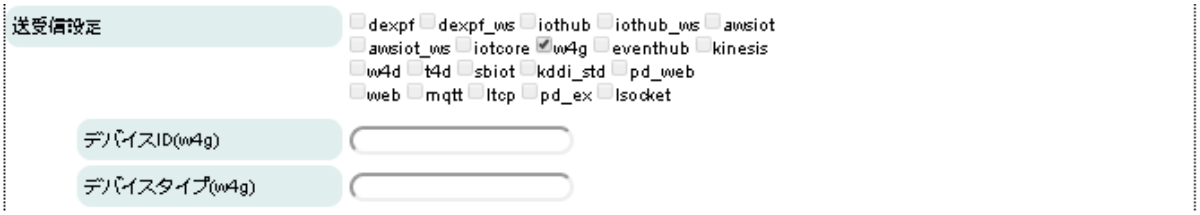


表 19: 各デバイス設定メニューにおける設定項目

設定項目	説明
デバイス ID(w4g)	Watson IoT for Gateway のデバイス ID を設定します。
デバイスタイプ (w4g)	Watson IoT for Gateway のデバイスタイプを設定します。

注釈:

- トラストストア・キーストア・プライベートキーは WEB UI の「システム」→「ファイル管理」タブにてアップロードしてください。

MS Azure Event hubs(eventhub)

■ 送受信設定メニューにおける設定

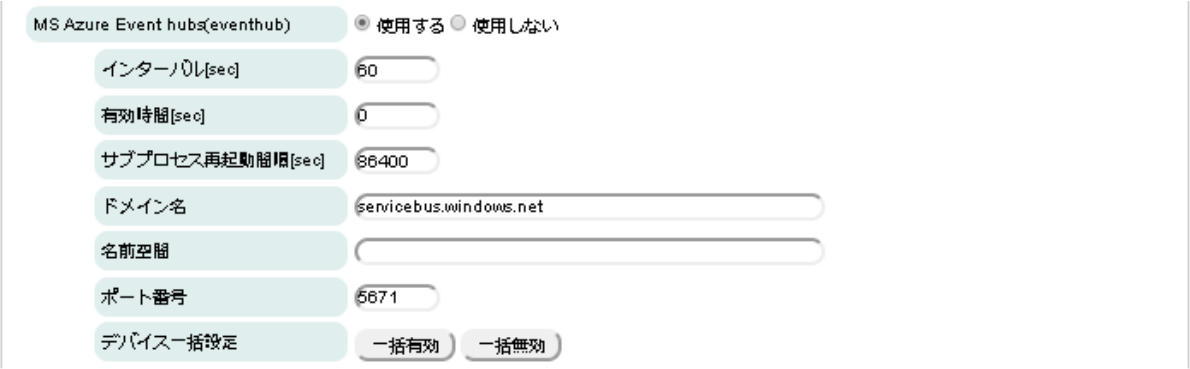


表 20: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ドメイン名	接続先の Event hubs のドメイン名を設定します。
名前空間	Event hubs の名前空間を設定します。
ポート番号	送信先の Event hubs のポート番号を設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

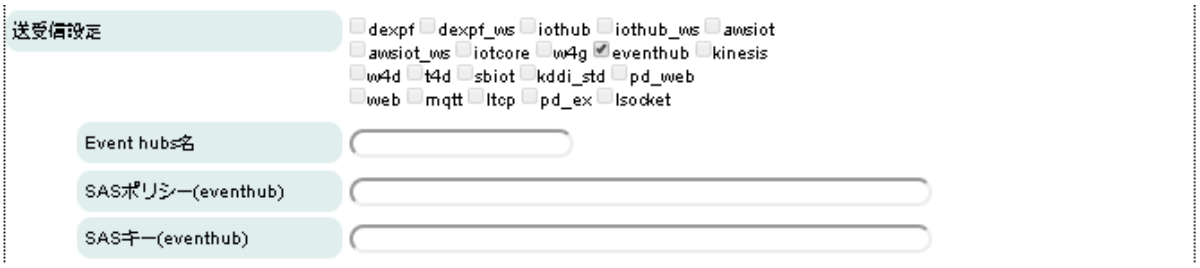


表 21: 各デバイス設定メニューにおける設定項目

設定項目	説明
Event hubs 名 (eventhub)	Event hubs の hub 名を設定します。
SAS ポリシー (eventhub)	Event hubs の SAS ポリシーを設定します。
SAS キー (eventhub)	Event hubs の SAS キーを設定します。

Amazon Kinesis(kinesis)

■ 送受信設定メニューにおける設定

表 22: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ドメイン名	送信先の Amazon Kinesis のドメイン名を設定します。
リージョン	送信先の Amazon Kinesis のリージョンコードを設定します。
アクセス ID	Amazon Kinesis のアクセス ID を設定します。
アクセスキー	Amazon Kinesis のアクセスキーを設定します。
ストリーム名	Amazon Kinesis のストリーム名を設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

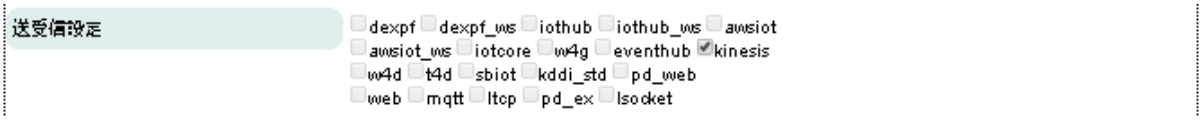


表 23: 各デバイス設定メニューにおける設定項目

設定項目	説明
	「送受信設定」をチェックする以外の設定項目はありません。

Watson IoT for Device(w4d)

■ 送受信設定メニューにおける設定

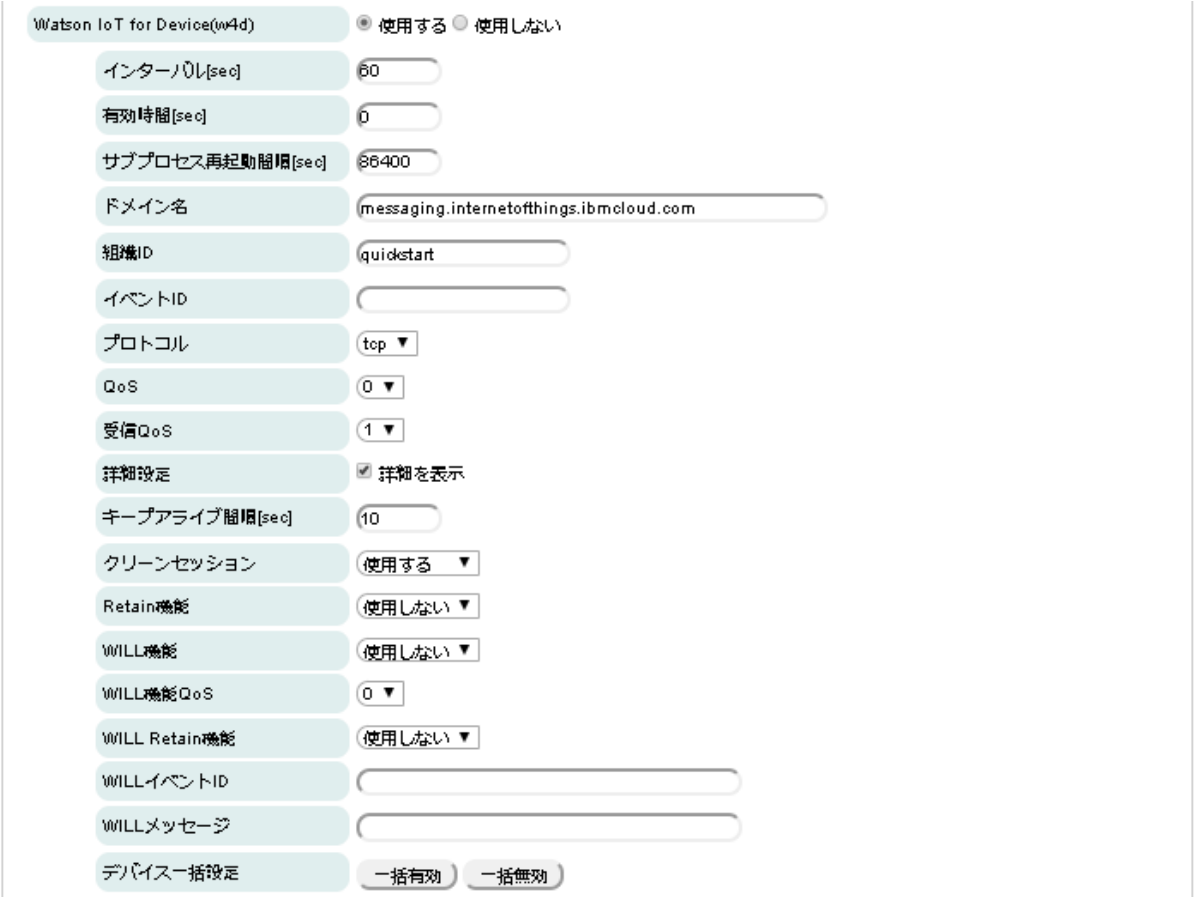


表 24: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ドメイン名	接続先の Watson IoT for Device のドメイン名を設定します。
組織 ID	Watson IoT for Device の組織 ID を設定します。 Quickstart を使用する場合には、“ quickstart ”を設定してください。
イベント ID	Watson IoT for Device のイベント ID を設定します。
プロトコル	接続に用いるプロトコル (TCP/SSL) を選択します。
QoS	Watson IoT for Device へデータを送信する際の MQTT プロトコルの QoS を設定します。 Quickstart を使用する場合には、0 に設定する必要があります。
受信 QoS	Watson IoT for Device から制御メッセージを受信する際の MQTT プロトコルの QoS を設定します。
トラストストア	Watson IoT for Device へ接続する際のルート証明書ファイルのパス名を設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
クリーンセッション	MQTT プロトコルのクリーンセッションの使用／不使用を選択します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
WILL 機能	MQTT プロトコルの LWT(LastWill and Testament) 機能の使用／不使用を選択します。
WILL QoS 機能	MQTT プロトコルが WILL メッセージを送出する際の QoS を設定します。
WILL Retain 機能	MQTT プロトコルの WILL Retain 機能の使用／不使用を選択します。
WELL イベント ID	MQTT の WILL トピックに用いられる Watson IoT for Device のイベント ID を設定します。
WLL メッセージ	MQTT プロトコルの WILL メッセージを設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

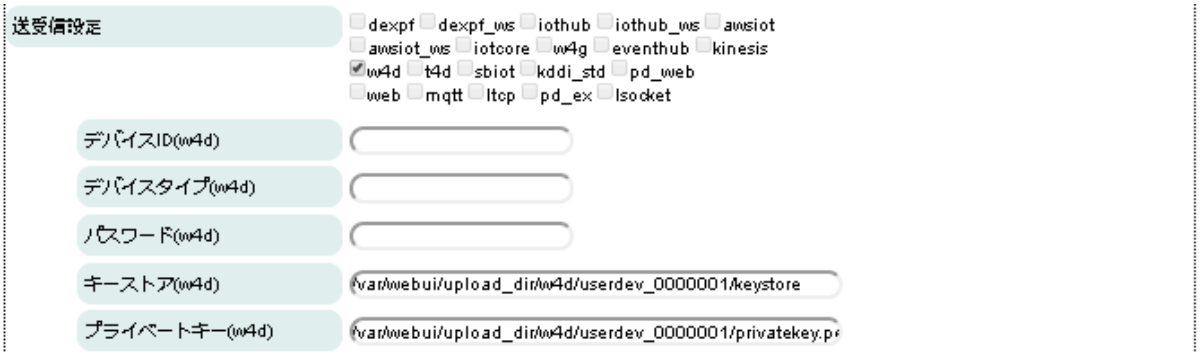


表 25: 各デバイス設定メニューにおける設定項目

設定項目	説明
デバイス ID(w4d)	Watson IoT for Device のデバイス ID を設定します。
デバイスタイプ (w4d)	Watson IoT for Device のデバイスタイプを設定します。
パスワード	Watson IoT for Device への接続に用いるパスワードを設定します。
キーストア	Watson IoT for Device へ接続する際のクライアント証明書ファイルのパス名を設定します。
プライベートキー	Watson IoT for Device へ接続する際のプライベートキーファイルのパス名を設定します。

注釈:

- トラストストア・キーストア・プライベートキーは WEB UI の「システム」→「ファイル管理」タブにてアップロードしてください。

Toami for DOCOMO(t4d)

■ 送受信設定メニューにおける設定

表 26: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
接続先 URL	接続先の Toami for DOCOMO の URL を設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

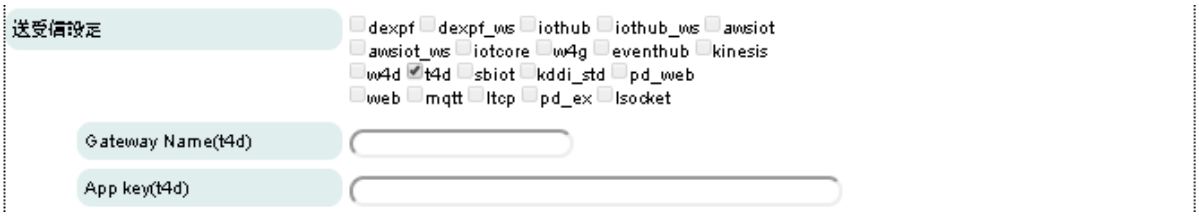
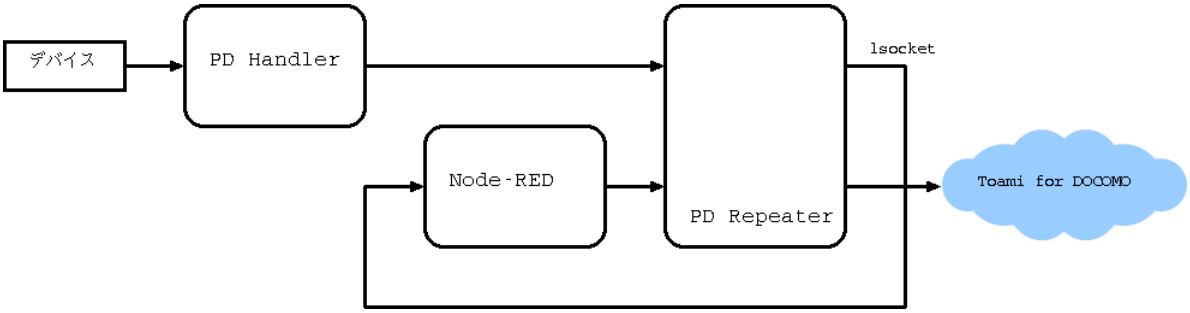


表 27: 各デバイス設定メニューにおける設定項目

設定項目	説明
Gateway Name(t4d)	Toami for DOCOMO の Gateway Name を設定します。
App Key(t4d)	Toami for DOCOMO の Gateway Name を設定します。

Toami for DOCOMO では扱えるデータ（JSON 文字列）のキー情報が固定されているため、下図の様に PD Handler の出力を PD Reperter で Node-RED に渡し、Node-RED でキー情報の変換を行い、PD Reperter を介して Toami for DOCOMO へ送る構成とする必要があります。

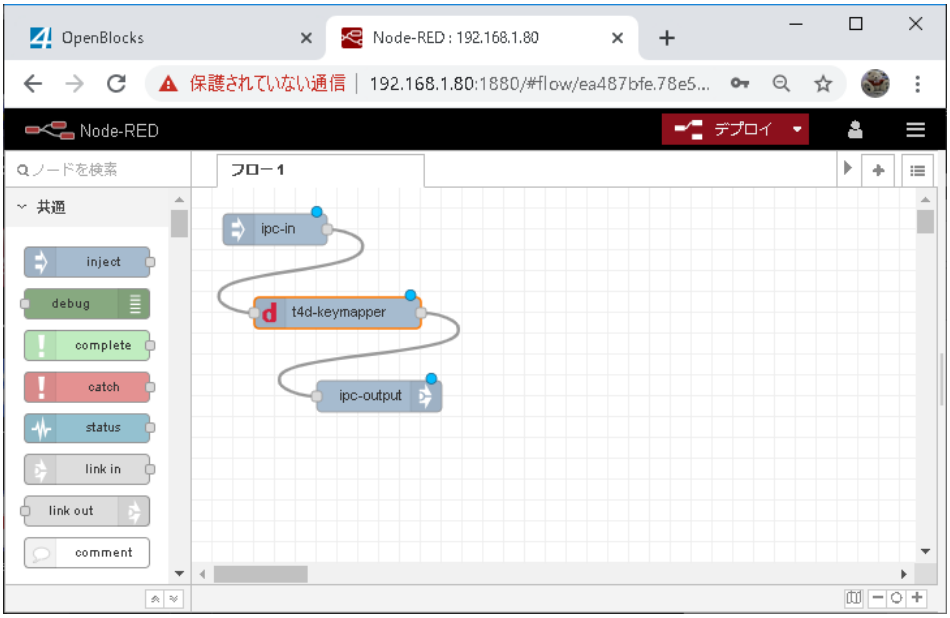


Beacon デバイス　（PD Handler BLE）を例に設定方法を示します。

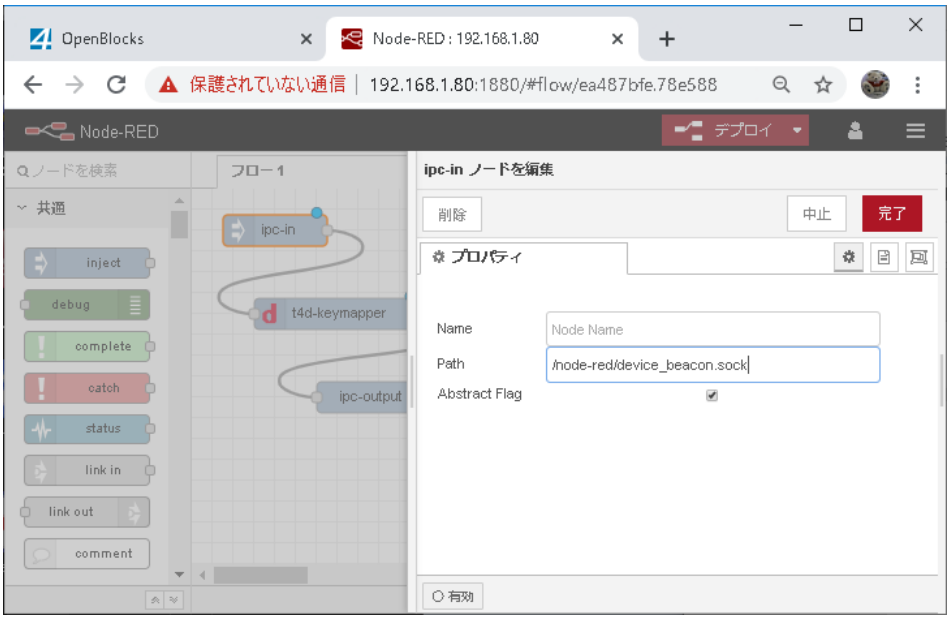
1. PD Handler BLE （デバイス番号 device_beacon）の送り先をドメインソケット (lsocket) に設定します。ドメインソケット (lsocket) の設定については、第 2-4-2-21 章を参照して下さい。ここで「ソケットパスプレフィックス」は、デフォルト値"@/node-red/"のまま設定されるものとします。

2. Node-RED をユーザー定義のデバイスとして登録します。ユーザー定義デバイスの登録については、第 2-4-13 章を参照して下さい。ここでユーザー定義のデバイスとして"userdev_0000001"が割り振られたものとします。

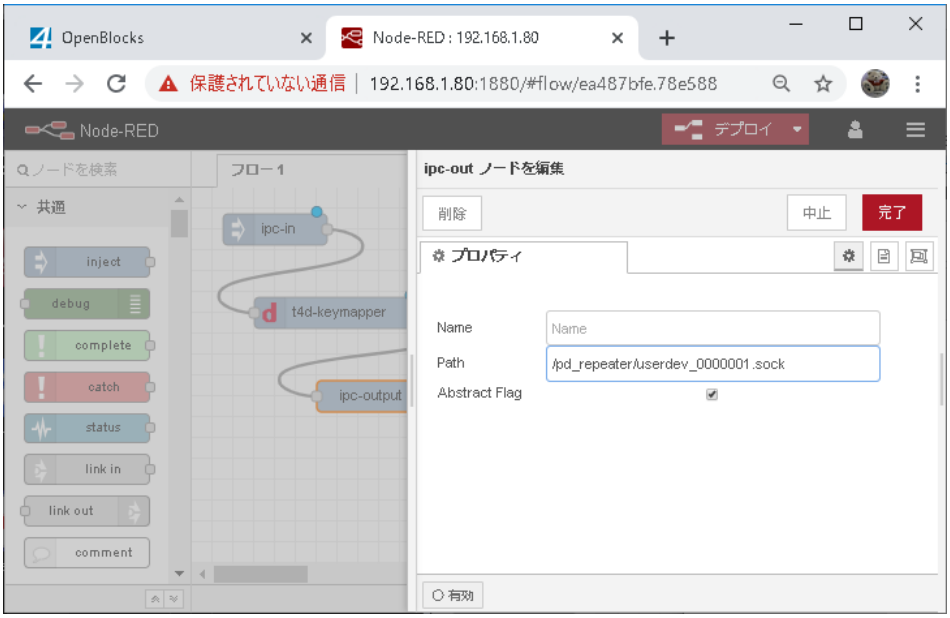
3. Node-RED において、ipc-in ノード・ipc-out ノード・t4d-keymapper ノードを配置し結線します。Node-RED の利用方法については、 *Node-RED* **スターターガイド** を参照して下さい。



4. ipc-in ノードのプロパティを設定します。プロパティの「Path」を"/node-red/device_beacon.sock"に設定し、「Abstract Flag」をチェックします。



5. ipc-out ノードのプロパティを設定します。プロパティの「Path」を"/pd_repeater/userdev_0000001.sock"に設定し、「Abstract Flag」をチェックします。



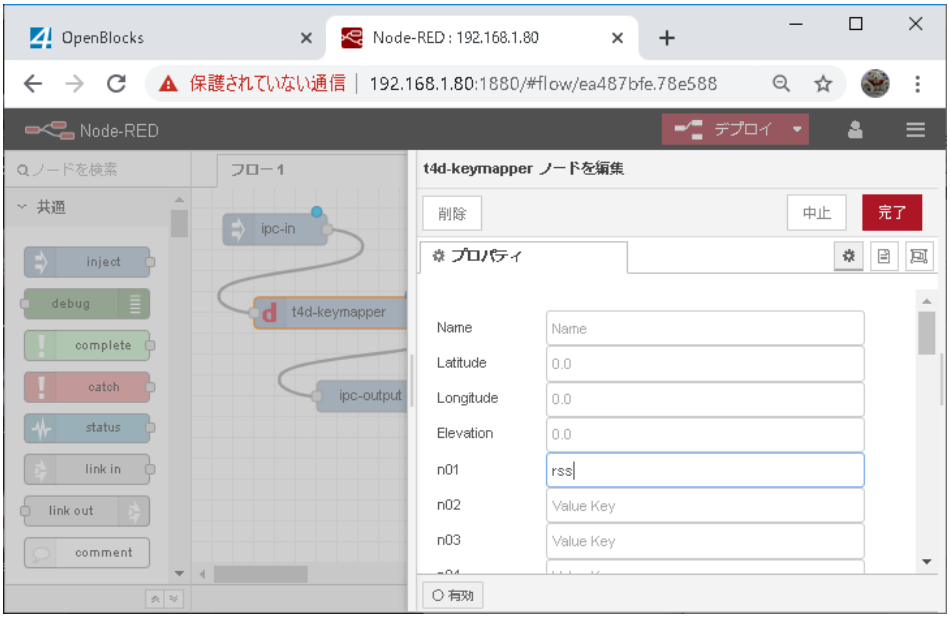
6. t4d-mapper ノードのプロパティにおいて PD Handler BLE のビーコン出力の JSON キーと変換後のキーの設定を行います。

表 28: PD Handler BLE のビーコン出力の JSON キーと変換後のキー

JSON キー	データ形式	内容	変換後のキー
deviceId	文字列	デバイス ID	s01
appendixInfo	文字列	付随情報	s02
time	文字列	データ取得時刻	s03
rssi	整数値	受信信号強度	n01
type	文字列	ビーコン種別	s04
data	文字列	ペイロードデータ（HEX 表記）	s05
status	文字列	ビーコンステータス	s06
memo	文字列	WEB UI から設定された文字列	s07

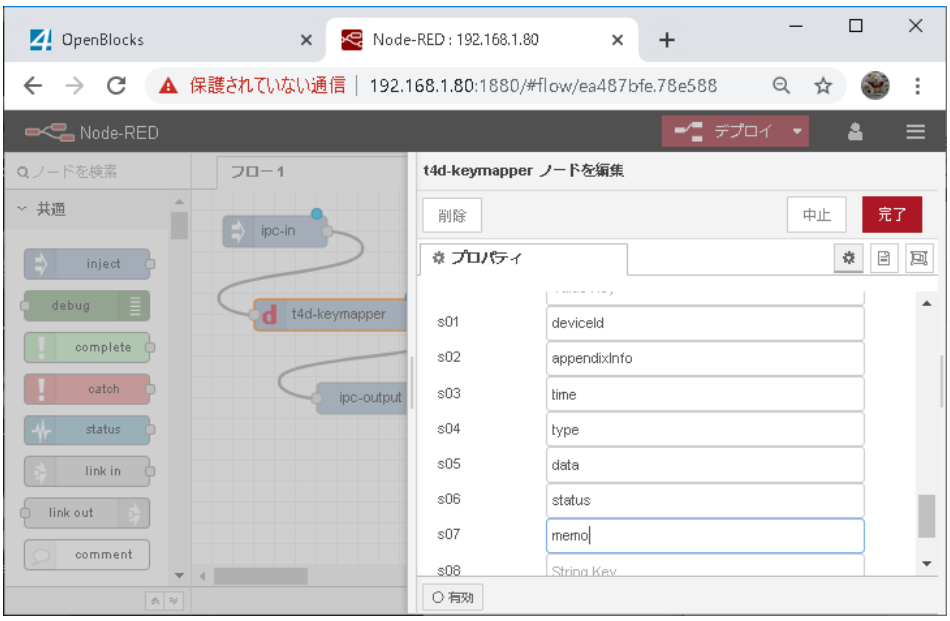
t4d-mapper ノードのプロパティ「n01」を設定します。

6-1. 「n01」を「rssi」に設定します。



t4d-mapper ノードのプロパティ「s01」〜s07」を設定します。

6-2. 「s01」を「deviceId」に設定します。
6-3. 「s02」を「appendixInfo」に設定します。
6-4. 「s03」を「time」に設定します。 *1
6-5. 「s04」を「type」に設定します。
6-6. 「s05」を「data」に設定します。
6-7. 「s06」を「status」に設定します。
6-8. 「s07」を「memo」に設定します。



注釈:

1. 「s03」に設定される「データ取得時刻」は、Toami for DOCOMO 上では単なる文字列として扱われます。Toami for DOCOMO の時刻情報（「timestamp」キーの値）は t4d-keymapper ノード内で自動的に生成され付加されます。

- t4d-keymapper ノードは、PD Handler Modbus Client/Server 等、値を配列や JSON オブジェクトで出力するデバイスには対応できません。値を配列や JSON オブジェクトで出力するデバイスについては、その変換条件に応じた独自の機能（ファンクション）ノードを作成して下さい。

SoftBank スマ可視専用クラウド (sbiot)

■ 送受信設定メニューにおける設定

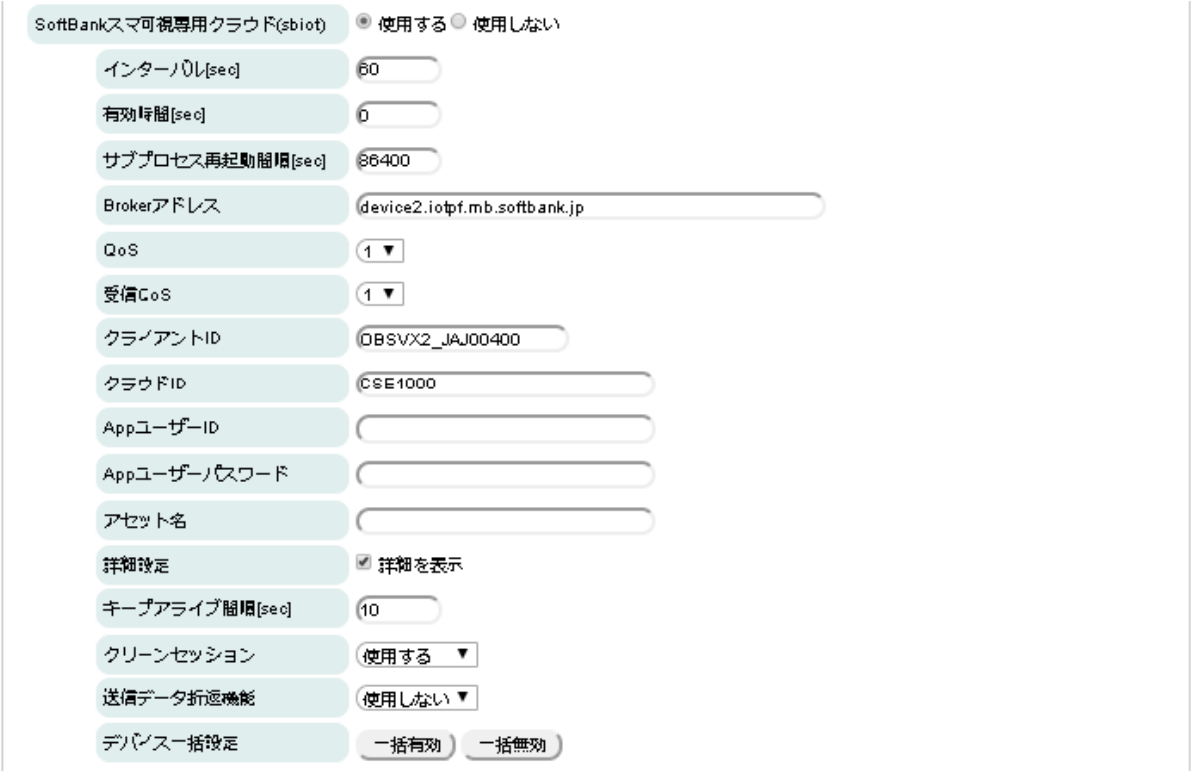


表 29: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
Broker アドレス	接続先の SoftBank スマぁ可視専用クラウドの Broker アドレスを設定します。
QoS	SoftBank スマ可視専用クラウドへデータを送信する際の MQTT プロトコルの QoS を設定します。
受信 QoS	SoftBank スマ可視専用クラウドから制御メッセージを受信する際の MQTT プロトコルの QoS を設定します。
クライアント ID	SoftBank スマ可視専用クラウドのクライアント ID を設定します。
クラウド ID	SoftBank スマ可視専用クラウドのクラウド ID を設定します。
App ユーザー ID	SoftBank スマ可視専用クラウドの APP ユーザー ID を設定します。
App ユーザーパスワード	SoftBank スマ可視専用クラウドの APP ユーザーパスワードを設定します。
アセット名	SoftBank スマ可視専用クラウドのアセット名を設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
クリーンセッション	MQTT プロトコルのクリーンセッションの使用／不使用を選択します。
送信データ折返機能	送信データの折り返しを受ける機能です。本機能は、本来データを送信する、あるいはデバイスに対する下流方向制御のメッセージを受信するデバイスゲートウェイを、データ受信するアプリケーションゲートウェイとして機能させるものです。下流方向制御が可能なデバイスを接続する場合は、データあるいはメッセージがループし無いよう配慮する必要があります。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

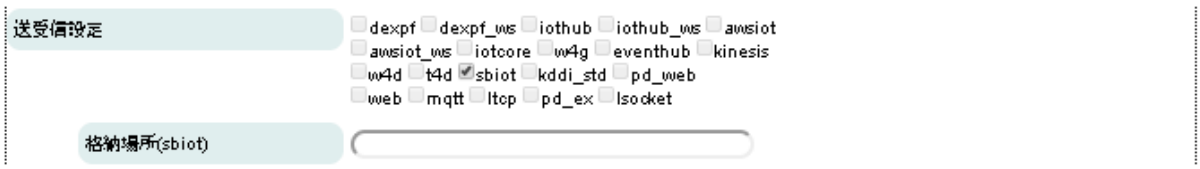


表 30: 各デバイス設定メニューにおける設定項目

設定項目	説明
格納場所 (sbiot)	SoftBank スマ可視専用クラウドの格納場所（コンテナ名）を設定します。

KDDI IoT クラウド Standard(kddi_std)

■ 送受信設定メニューにおける設定

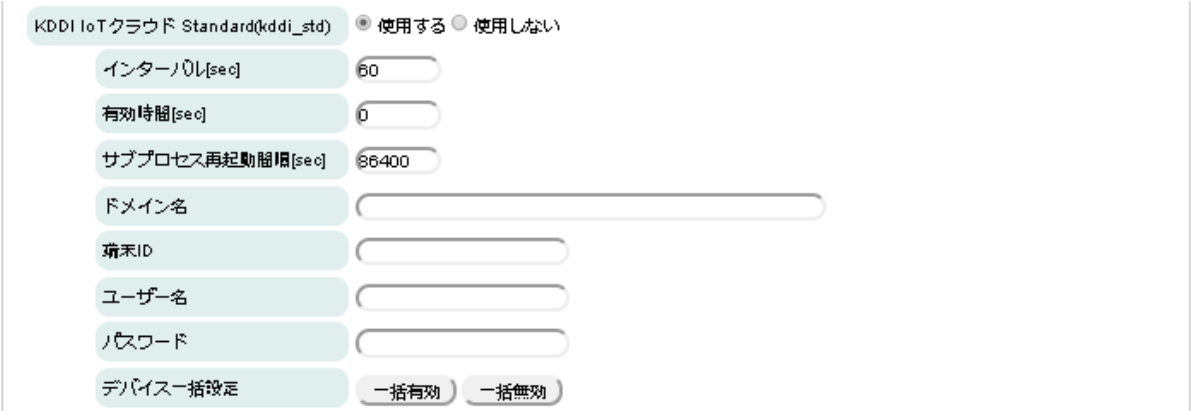


表 31: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ドメイン名	接続先の KDDI IoT クラウド Standard のドメイン名を設定します。
端末 ID	KDDI IoT クラウド Standard の端末 ID を設定します。
ユーザー名	KDDI IoT クラウド Standard の BASIC 認証に用いるユーザー名を設定します。
パスワード	KDDI IoT クラウド Standard の BASIC 認証に用いるパスワードを設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

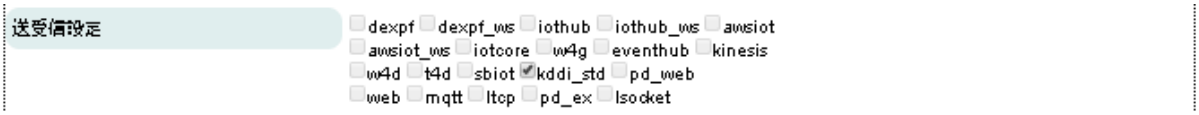


表 32: 各デバイス設定メニューにおける設定項目

設定項目	説明
	「送受信設定」をチェックする以外の設定項目はありません。

PH 社独自仕様 WEB サーバー (pd_web)

■ 送受信設定メニューにおける設定

表 33: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
接続先 URL	接続先の PH 社独自仕様 WEB サーバーの URL を設定します。
受信ポーリング間隔 [sec]	WEB サーバーから制御メッセージを読み出す間隔を設定します。
ユーザー名	WEB サーバーの BASIC 認証に用いるユーザー名を設定します。
パスワード	WEB サーバーの BASIC 認証に用いるユーザー名を設定します。
最大 POST データサーズ	1 回の POST メソッドで送信する最大データサイズを選択します。1~4Mbyte の中で選択します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

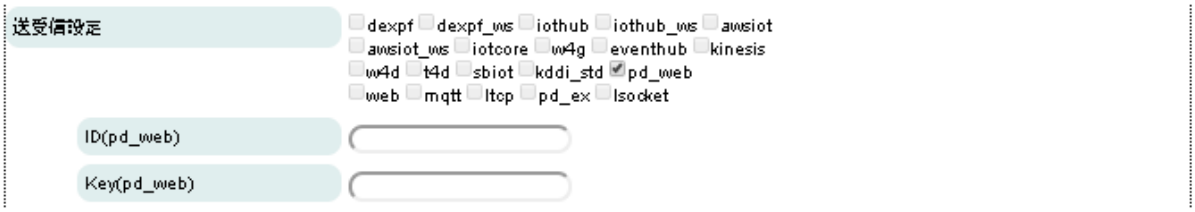


表 34: 各デバイス設定メニューにおける設定項目

設定項目	説明
ID(pd_web)	WEB サーバーのトークン認証に用いる ID を設定します。
Key(pd_web)	WEB サーバーのトークン認証に用いる Key を設定します。

注釈:

- PH 社独自仕様 WEB サーバーでのデータ送信方法は Content-Type を "application/json"として POST しています。
- 別途 HTTP ヘッダー等を付与して送信していますので、詳細仕様については *PH* **社独自仕様** *Web* **サーバー（***PD Web***）** を参照して下さい。

WEB サーバー (web)

■ 送受信設定メニューにおける設定

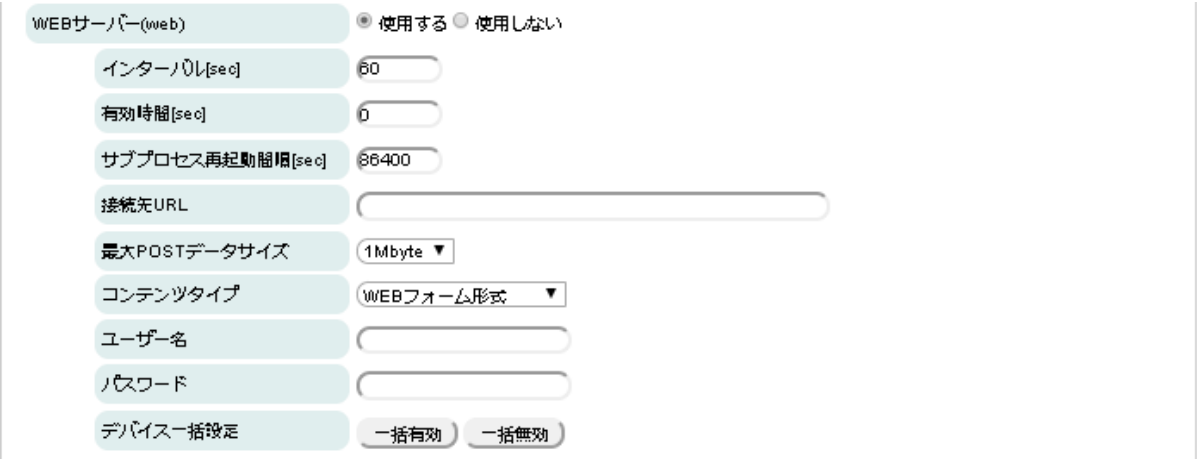


表 35: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
接続先 URL	接続先の汎用 WEB サーバーの URL を設定します。
最大 POST データサーズ	1 回の POST メソッドで送信する最大データサイズを選択します。1~4Mbyte の中で選択します。
コンテンツタイプ	WEB サーバーのコンテンツタイプ（WEB フォーム形式 *1 ／プレーンテキスト形式 *2 ／ JSON 形式 *3）を選択します。
ユーザー名	WEB サーバーの BASIC 認証に用いるユーザー名を設定します。
パスワード	WEB サーバーの BASIC 認証に用いるユーザー名を設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

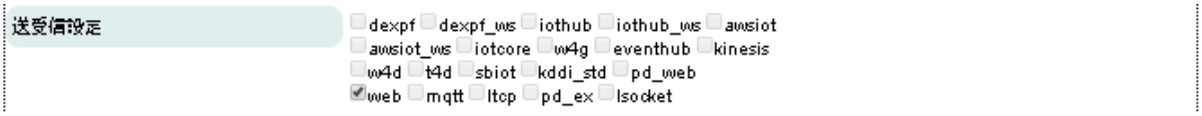


表 36: 各デバイス設定メニューにおける設定項目

設定項目	説明
	「送受信設定」をチェックする以外の設定項目はありません。

注釈:

- WEB サーバに対しては、データを POST メソッドにて送信します。

1. 送信形式設定が"WEB フォーム形式"の場合、Content-Type は "application/x-www-form-urlencoded"となります。ペイロード（送信データ本体）は、"Records" という x-www-form-urlencoded 変数に複数データをまとめて送信します。
2. 送信形式設定が"プレーンテキスト形式"の場合、Content-Type は "text/plain"となります。ペイロード（送信データ本体）の URL セーフエンコードは行われず、複数データ (JSON 前提) をまとめて送信します。
3. 送信形式設定が"JSON 形式"の場合、Content-Type は "application/json"となります。ペイロード（送信データ本体）の URL セーフエンコードは行われず、複数データ (JSON 前提) をまとめて送信します。

1. 送信形式設定 : WEB フォーム形式

- データ書式

```
Records = [{DATA1},{DATA2},{DATA3},…{DATAn}]
```

- 送信サンプル

```
POST / HTTP/1.0
Content-Length: 422
Content-Type: application/x-www-form-urlencoded

Records=[
{
 "deviceId": "b0b448b81105", "memo": "cc2650-1", "objectTemp": 20, "ambientTemp
↪": 24.84375,
 "humidity":4 7.91259765625, "temperature": 25.32928466796875, "pressure":␣
↪1016.37,
 "time":"2016-11-24T16:50:23.431+0900"
 },
 {
  "deviceId": "b0b448b81105", "memo": "cc2650-1", "objectTemp": 19.75,
↪"ambientTemp":24.875,
  "humidity": 47.91259765625, "temperature": 25.3594970703125, "pressure":1016.
↪31,
  "time": "2016-11-24T16:50:26.459+0900", "lux": 427.68
 }
]
```

2. 送信形式設定 : プレーンテキスト形式

- データ書式

```
[{DATA1}, {DATA2}, {DATA3}, … {DATAn}]
```

- 送信サンプル

```
POST / HTTP/1.0
Content-Type: text/plain
Content-Length: 209

[
 {
  "deviceId": "6d4adcb7133f", "appendixInfo": "G8H00010", "rssi": -90,
  "time": "2017-12-04T14:07:18.157+09:00"
 },
 {
  "deviceId": "e135535f61e4", "appendixInfo": "G8H00010", "rssi": -74,
  "time": "2017-12-04T14:07:18.762+09:00"
 }
]
```

3. 送信形式設定 : JSON 形式

- データ書式

```
[{DATA1}, {DATA2}, {DATA3}, … {DATAn}]
```

- 送信サンプル

```
POST / HTTP/1.0
Content-Type: application/json
Content-Length: 209

[
 {
  "deviceId": "6d4adcb7133f", "appendixInfo": "G8H00010", "rssi": -90,
  "time": "2017-12-04T14:07:18.157+09:00"
 },
 {
  "deviceId": "e135535f61e4", "appendixInfo": "G8H00010", "rssi": -74,
  "time": "2017-12-04T14:07:18.762+09:00"
 }
]
```

MQTT サーバー (mqtt)

■ 送受信設定メニューにおける設定

表 37: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
送信先ホスト	接続先の汎用 MQTT サーバーのホスト名を設定します。
クライアント ID	汎用 MQTT サーバーに接続するためのクライアント ID を設定します。
トピックプレフィックス	汎用 MQTT サーバーへデータを送る際に用いる MQTT プロトコルのトピックのプレフィックスを設定します。サフィックスは ’/ ’に続きユニーク ID が設定されます。
受信トピックプレフィックス	汎用 MQTT サーバーから制御メッセージを待ち受けるために用いる MQTT プロトコルのトピックのプレフィックスを設定します。サフィックスは ’/ ’に続きユニーク ID が設定されます。
プロトコル	接続に用いるプロトコル (TCP/SSL) を選択します。
ポート番号	送信先ホストのポート番号を設定します。
QoS	汎用 MQTT サーバーへデータを送信する際の MQTT プロトコルの QoS を設定します。
受信 QoS	汎用 MQTT サーバーから制御メッセージを受信する際の MQTT プロトコルの QoS を設定します。
ユーザー名	汎用 MQTT サーバーへの接続に用いるユーザー名を設定します。
パスワード	汎用 MQTT サーバーへの接続に用いるパスワードを設定します。
トラストストア	汎用 MQTT サーバーへ接続する際のルート証明書ファイルのパス名を設定します。
キーストア	汎用 MQTT サーバーへ接続する際のクライアント証明書ファイルのパス名を設定します。
プライベートキー	汎用 MQTT サーバーへ接続する際のプライベートキーファイルのパス名を設定します。
詳細設定	MQTT プロトコルの詳細値を設定します。
キープアライブ間隔 [sec]	MQTT プロトコルのキープアライブインターバルを設定します。
クリーンセッション	MQTT プロトコルのクリーンセッションの使用／不使用を選択します。
Retain 機能	MQTT プロトコルの Retain 機能の使用／不使用を選択します。
WILL 機能	MQTT プロトコルの LWT(LastWill and Testament) 機能の使用／不使用を選択します。
WILL QoS 機能	MQTT プロトコルが WILL メッセージを送出する際の QoS を設定します。
WILL Retain 機能	MQTT プロトコルの WILL Retain 機能の使用／不使用を選択します。
WELL トピックサフィックス	MQTT の WILL トピックのサフックスを設定します。WILL トピックのプレフックス部には、トピックプレフィックスに設定された文字列が用いられます。
WLL メッセージ	MQTT プロトコルの WILL メッセージを設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

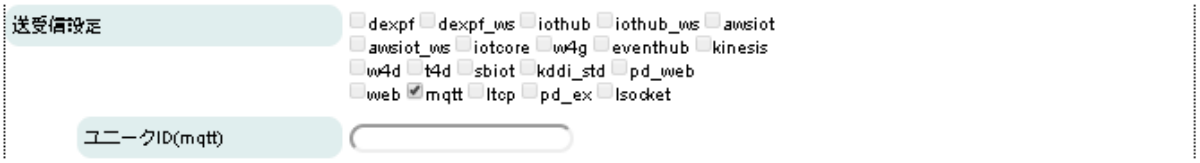


表 38: 各デバイス設定メニューにおける設定項目

設定項目	説明
ユニーク ID(mqtt)	汎用 MQTT サーバーへデータを送る際、もしくは制御メッセージを受け取る際の MQTT プロトコルのトピックのサフィックスに用いる各デバイス毎にユニークな文字列を設定します。'. '（ASCII コード 0x2e）は '/ ' (ASCII コード 0x2f) に変換して扱われます。

注釈:

- トラストストア・キーストア・プライベートキーは WEB UI の「システム」→「ファイル管理」タブにてアップロードしてください。

TCP(ltcp)

■ 送受信設定メニューにおける設定

表 39: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
IP アドレス	接続先の汎用 TCP サーバーの IP アドレスを設定します。
デリミタ	データーの境界となる 1 文字もしくは 2 文字の ASCII コードを設定します。例えば CR（復帰）をデリミタとする場合は 0x0d、CRLF（復帰・改行）をデリミタとする場合は 0x0d0a となります。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

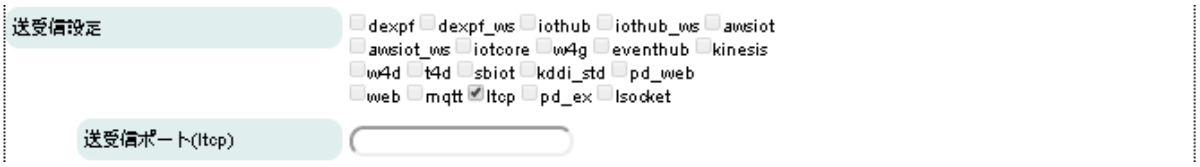


表 40: 各デバイス設定メニューにおける設定項目

設定項目	説明
送受ポート番号 (ltcp)	接続先の汎用 TCP サーバーのポート番号を 49152〜65535 の範囲で設定します。

PD Exchange(pd_ex)

■ 送受信設定メニューにおける設定

表 41: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
接続先 URL	接続先の PD Exchange の URL を設定します。
ポーリング間隔 [sec]	PD Exchange から制御メッセージを読み出す間隔を設定します。
シークレットキー	PD Exchange のアカウントのシークレットキーを設定します。
デバイス ID プレフィックス	PD Exchange のアカウントに対するデバイス ID プレフィックスを設定します。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

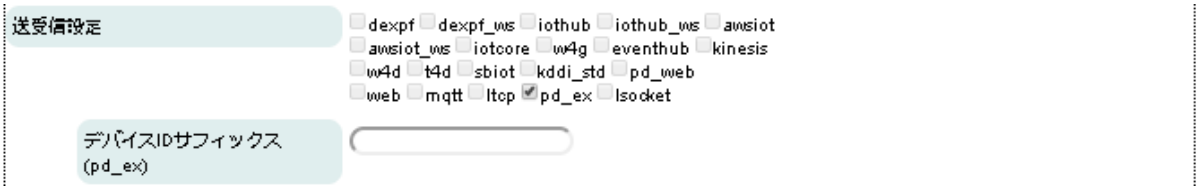


表 42: 各デバイス設定メニューにおける設定項目

設定項目	説明
デバイス ID サフィックス (pd_ex)	PD Exchange のデバイス ID のサフィックスを設定します。

ドメインソケット (lsocket)

■ 送受信設定メニューにおける設定

表 43: 送受信設定メニューにおける設定項目

設定項目	説明
インターバル [sec]	送信完了後〜送信開始までの時間間隔を秒単位で設定します。
有効時間 [sec]	PD Reperter がデータ送信できない場合において、保持する時間を設定します。0 を指定した場合、データ送信が完了するまで保持し続けます。
サブプロセス再起動間隔 [sec]	サブプロセスを再起動する間隔を設定します。通常デォルト値（86400 秒）から変更する必要はありません。0 を指定した場合、再起動は行いません。
ソケットパスプレフィックス	データを出力する UNIX ドメインソケットのパス名のプレフィックスを設定します。先頭が ’@ ’の場合は Abstract Domain Socket として扱われます。パス名のサフィックスにはデバイス番号が設定されます。
デバイス一括設定	各デバイス設定メニューにおいて送信対象設定が「送信する」となっている各対象の送信先設定を一括で有効/無効を選択できます。

■ 各デバイス設定メニューにおける設定

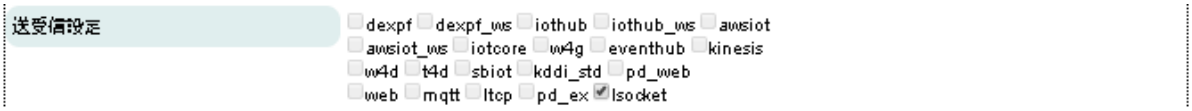


表 44: 各デバイス設定メニューにおける設定項目

設定項目	説明
	「送受信設定」をチェックする以外の設定項目はありません。

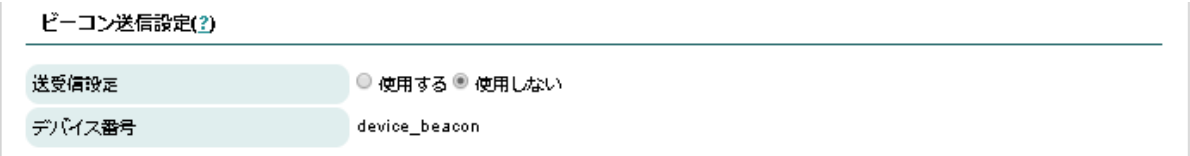
20.1.2 ビーコン送信設定

WEB UI の「IoT データ」→「送受信設定」タブの「ビーコン送信設定」メニューより、ビーコンの送信設定を行うことが出来ます。

ビーコンの送信を行うためには、WEB UI の「サービス」→「基本」→「BT I/F」タブより、サービス機能の基本設定とし BT I/F (hci0) が「使用する」に設定されている必要があります。

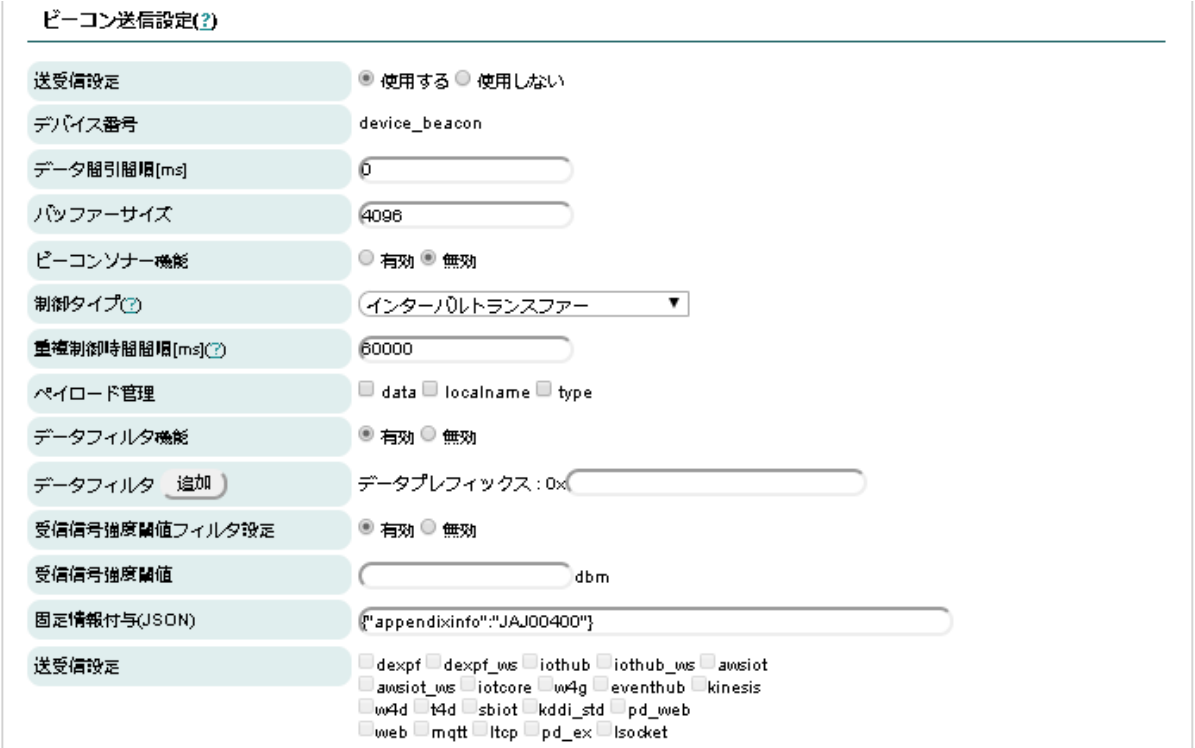
サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

「ビーコン送信設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler BLE を「使用する」に設定して下さい。



初期状態の送信先設定は上図のようになっています。ここで、ビーコンデータをクラウド等への送信する場合には、"送信する"を選択します。

> **警告:** 後述のデバイス情報送信設定で送信対象としているビーコンには、本項は適用されません。



「送信する」を選択すると各設定項目が上図のように表示されます。

表 45: ビーコン送信設定の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
ビーコンソナー機能	受信対象となっているビーコンデータを受信した際にビーコンソナーを有効にするか無効を設定します。 *1
制御タイプ	ビーコンデータを管理する方式を以下から選択します。各方式については後述の"ビーコン重複制御アルゴリズム"を参照してください。 1. インターバルトランスファー 2. エントリーポイントトランスファー 3. インアウトステータストランスファー
複制御時間間隔 [ms]	各制御タイプにて用いる制御時間を msec 単位で設定します。
ペイロード管理	ビーコンデータを PD Repeater へ渡す際に、ビーコンの各情報を付随させるかを選択します。 **data:** アドバタイズデータ (16 進数) **localname:** デバイス名 **type:** データ種別
データフィルタ機能	(データプレフィックス) 送信対象のビーコンを選別するフィルタを設定します。データプレフィックスに 16 進文字列でフィルタ条件を入力すると、ビーコンのアドバタイズ情報を前方一致で比較し一致したもののみを送信先へ送信します。 • 「追加」ボタンにて、複数登録できます。 • データフィルタを設定する場合には、本装置内 (local) 内のログの data を参照しデバイスをフィルタリングしてください。　本装置内のログは (local) 内のログについてもフィルタは適用されます。
受信信号強度閾値フィルタ設定	受信対象とするビーコンの信号強度閾値フィルタを使用するか設定します。
受信信号強度閾値	受信対象とするビーコンの信号強度を設定します。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
送信先設定	“ 使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

注釈:

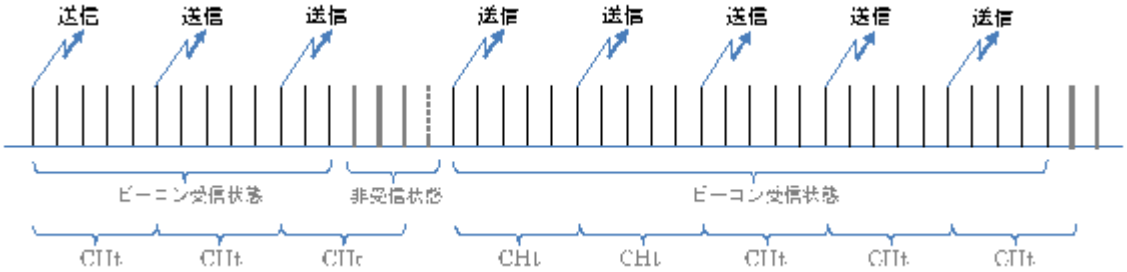
1. ビーコンソナー機能を送信対象にした状態において USB スピーカー (型番：MM-SPU8BK) を接続した状態にて受信対象 (データフィルタ及び受信信号強度閾値フィルタについても考慮) となっているビーコンデータを受信した場合には、スピーカーから検出音が鳴ります。

- C 言語版 PD Handler BLE で対応していない非コネクション型 BLE センサーについては、*Lua* **拡張**機能で対応することが可能です。

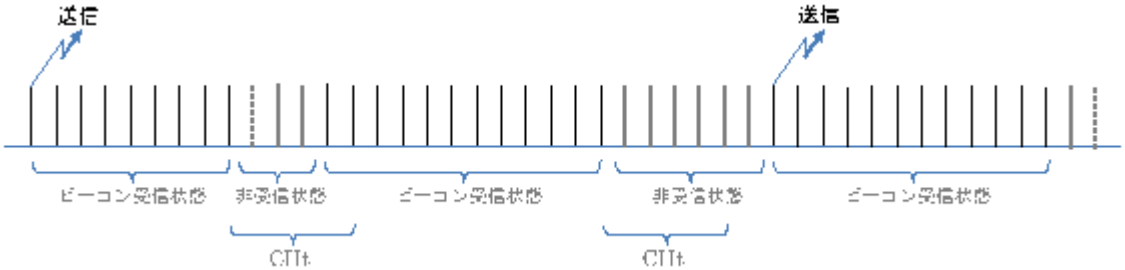
■ ビーコン重複制御アルゴリズム

- この説明における前提となる設定
  - ビーコンの送信間隔 ＝ 1 秒
  - 重複制御時間間隔（CHt）＝ 5 秒

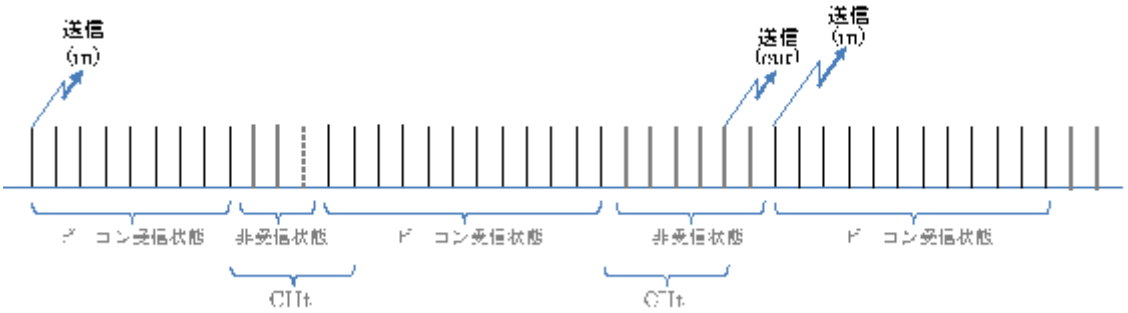
1. インターバルトランスファー
   ビーコンを受信している間は、指定された一定間隔で送信プログラムへ。



2. エントリーポイントトランスファー
   （CHt 時間内の一時非受信は退場扱いとしない）



3. インアウトステータストランスファー
   （CHt 時間内の一時非受信は退場扱いとしない）



20.1.3 BLE デバイス情報送信設定

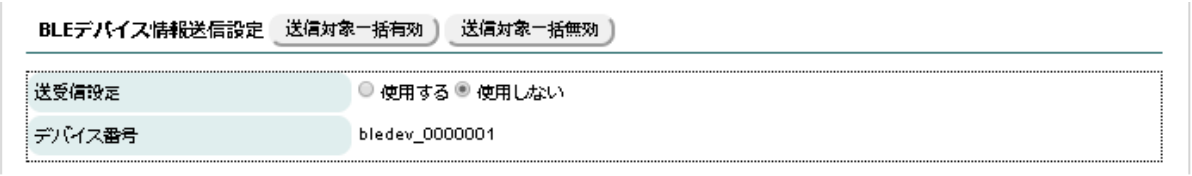
WEB UI の「IoT データ」→「送受信設定」タブの「BLE デバイス情報送信設定」メニューより、BLE デバイス情報の送信設定を行うことが出来ます。

BLE デバイス情報の送信を行うためには、WEB UI の「サービス」→「基本」→「BT I/F」タブより、サービス機能の基本設定として BT I/F (hci0) が「使用する」に設定されている必要があります。

また、WEB UI の「サービス」→「基本」→「BLE 登録」タブより、サービス機能の基本設定として BLE デバイスが登録されている必要があります。

サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

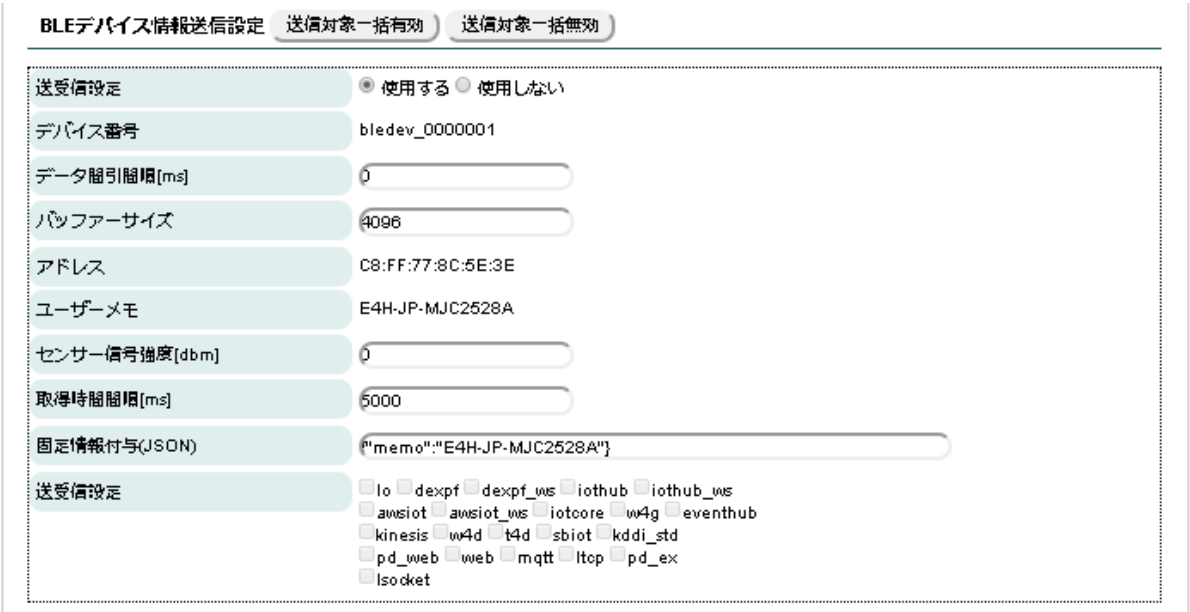
「BLE デバイス情報送信設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler BLE を「使用する」に設定して下さい。



登録済の BLE デバイスが存在している場合、初期状態では上図のようになっています。(デバイスが 1 個登録されている場合です。)

デバイス毎に送信対象項目にて「送信する」を選択すると各設定項目が表示されます。

送信対象一括有効、送信対象一括無効ボタンにて全ての登録済のデバイスの送信対象を制御できます。



「送信する」を選択すると各設定項目が上図のように表示されます。

表 46: BLE デバイス情報送信設定の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
アドレス	登録されたデバイスの BT のアドレスを表示します。
ユーザーメモ	登録されたデバイスにて設定されたメモ情報を表示します。
センサー信号強度 [dbm]	センサーに信号強度を設定できる機種の場合、設定したい信号強度を入力します。設定した信号強度が無い場合、近似値またはデフォルト値が設定されます。
取得時間間隔 [ms]	センサーからデータを取得する時間間隔を数字で設定します。単位は msec です。 *1
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

注釈:

1. 富士通コンポーネント製 BT Smart センサービーコン"と"Texas Instruments 製 SimpleLink SensorTag" のみサポートしています。

- 既存のデバイス不良等の差し替え時に以前のものと同様に扱う為に設定を同一にすることを推奨します。（不良となったデバイスは送信対象設定を「送信しない」へ変更してください。）

20.1.4 EnOcean デバイス設定

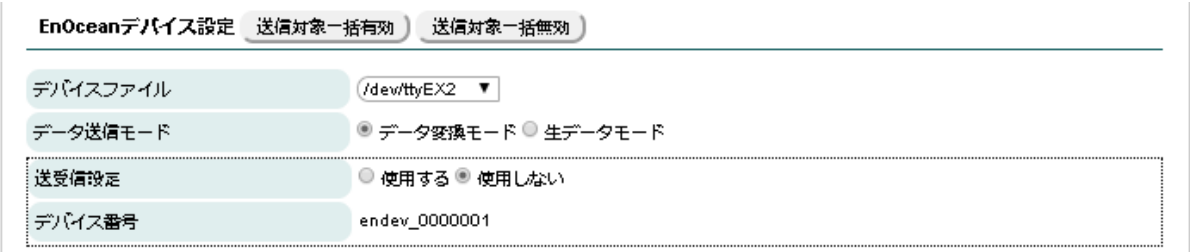
OpenBlocks シリーズに EnOcean モジュール（拡張追加モジュール）を搭載する、もしくは EnOcean 受信用 USB ドングルを装着することで、PD Handler UART を用いて EnOcean デバイスから情報を取得することができます。

WEB UI の「IoT データ」→「送受信設定」タブの「EnOcean デバイス設定」メニューより、EnOcean デバイスから情報を取得し送信する設定を行うことが出来ます。

EnOcean デバイスから情報を取得するためには、WEB UI の「サービス」→「基本」→「EnOcean 登録」タブより、サービス機能の基本設定として EnOcean デバイスが登録されている必要があります。

サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

「EnOcean デバイス設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler UART を「使用する」に設定し「使用対象」として「EnOcean」を選択して下さい。



登録済の EnOcean デバイスが存在している場合、初期状態では上図のようになっています。(デバイスが 1 個登録されている場合です。)

表 47: EnOcean デバイス設定の初期設定項目

設定項目	説明
デバイスファイル	拡張追加モジュールを使用する場合は「/dev/ttyEX2」、USB ドングルを使用する場合は対応するデバイスファイルを選択して下さい。 *1 • USB ドングルの場合、USB デバイスの認識順によってデバイスファイル名が異なります。
データ送信モード	PD Repeate へ送信するデータのモードを設定します。データ変換モードは対応している EEP の場合は解析したデータを PD Repeater へ送信します。対応していない EEP の場合は、受信データを 16 進数文字列へ変換したデータを PD Repeater へ送信します。また、生データモードは対応 EEP を問わず、受信データを 16 進数文字列へ変換したデータを PD Repeater へ送信します。

デバイス毎に送信対象項目にて「送信する」を選択すると各設定項目が表示されます。

送信対象一括有効、送信対象一括無効ボタンにて全ての登録済のデバイスの送信対象を制御できます。

「送信する」を選択すると各設定項目が上図のように表示されます。

表 48: EnOcean デバイス設定の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
デバイス ID	登録された EnOcean デバイスの ID を表示します。
EEP（機器情報プロファイル）	登録された EnOcean デバイスの機器情報プロファイルを表示します。
ユーザーメモ	登録されたデバイスにて設定されたメモ情報を表示します。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
送信先設定	"使用する ”を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

注釈:

1. USB ドングルの場合、USB デバイスの認識順によってデバイスファイル名が異なります。

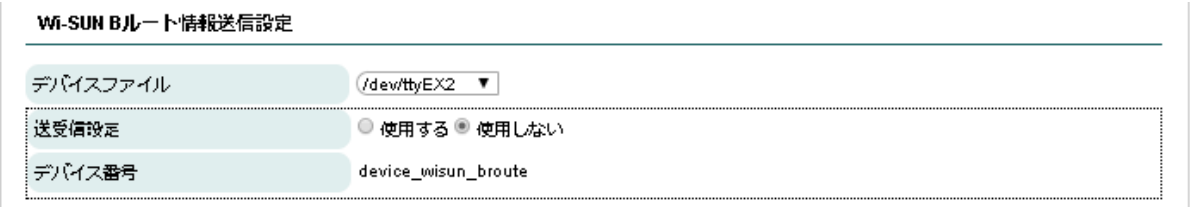
- 既存のデバイス不良等の差し替え時に以前のものと同様に扱う為に設定を同一にすることを推奨します。（不良となったデバイスは送信対象設定を「送信しない」へ変更してください。）
- 未対応の EnOcean デバイスについては、*Lua* **拡張** 機能で対応することが可能です。

20.1.5 Wi-SUN B ルート情報送信設定

OpenBlocks シリーズに Wi-SUN B ルートデバイス（拡張追加モジュール）を搭載することで、PD Handler UART を用いて Wi-SUN B ルートデバイスから情報を取得することができます。

WEB UI の「IoT データ」→「送受信設定」タブの「Wi-SUN B ルート情報送信設定」メニューより、Wi-SUN B ルートデバイスから情報を取得し送信する設定を行うことが出来ます。

「Wi-SUN B ルート情報送信設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler UART を「使用する」に設定し「使用対象」として「Wi-SUN B ルート」を選択して下さい。



初期状態の送信先設定は上図のようになっています。

表 49: EnOcean デバイス設定の初期設定項目

設定項目	説明
デバイスファイル	拡張追加モジュールを使用する場合は「/dev/ttyEX2」を選択して下さい。

「送信する」を選択すると各設定項目が上図のように表示されます。

表 50: ビーコン送信設定の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
B ルート ID	スマートメーターに接続する際の B ルート ID を設定して下さい。B ルート ID は "00" から始まります。
B ルートパスワード	スマートメーターに接続する際のパスワードを設定して下さい。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

20.1.6 Modbus クライアントデバイス設定

Modbus クライアントは、Modbus プロトコルを用いて PLC 機器から定期的にデータを読み込み（ポーリングを行い)、PD Repeater を介してクラウドに送ります。

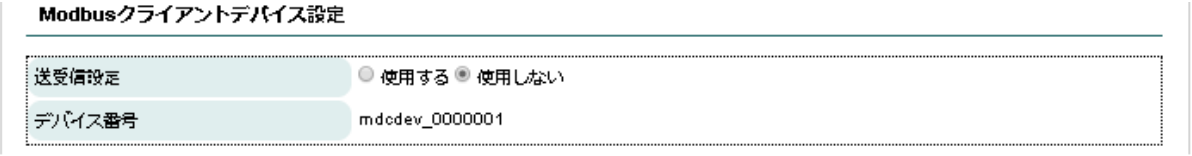
また、PD Repeater を介してクラウドから送られる制御メッセージ (JSON 文字列) に基づき PLC 機器に Modbus プロトコルを用いて接続しデータを読み書きすることもできます。

WEB UI の「IoT データ」→「送受信設定」タブの「Modbus クライアントデバイス設定」メニューより、Modbus クライアントデバイスの設定を行うことが出来ます。

Modbus クライアントデバイスを利用するためには、WEB UI の「サービス」→「基本」→「Modbus(C) 登録」タブより、サービス機能の基本設定として Modbus クライアントデバイスが登録されている必要があります。

サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

「Modbus クライアントデバイス設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler MODBUS Client を「使用する」に設定して下さい。



登録済の Modbus クライアントデバイスが存在している場合、初期状態では上図のようになっています。(デバイスが 1 個登録されている場合です。)

Modbus クライアントデバイスとは、PLC 機器そのものではなく、対象となる PLC 機器への接続方法の他、データを取得するための「読込方法」や「読込開始アドレス」、「読込レジスタ数」等の設定の組み合わせを意図します。

デバイス毎に送信対象項目にて「送信する」を選択すると各設定項目が表示されます。

送信対象一括有効、送信対象一括無効ボタンにて全ての登録済のデバイスの送信対象を制御できます。

「送信する」を選択すると各設定項目が次のように表示されます。

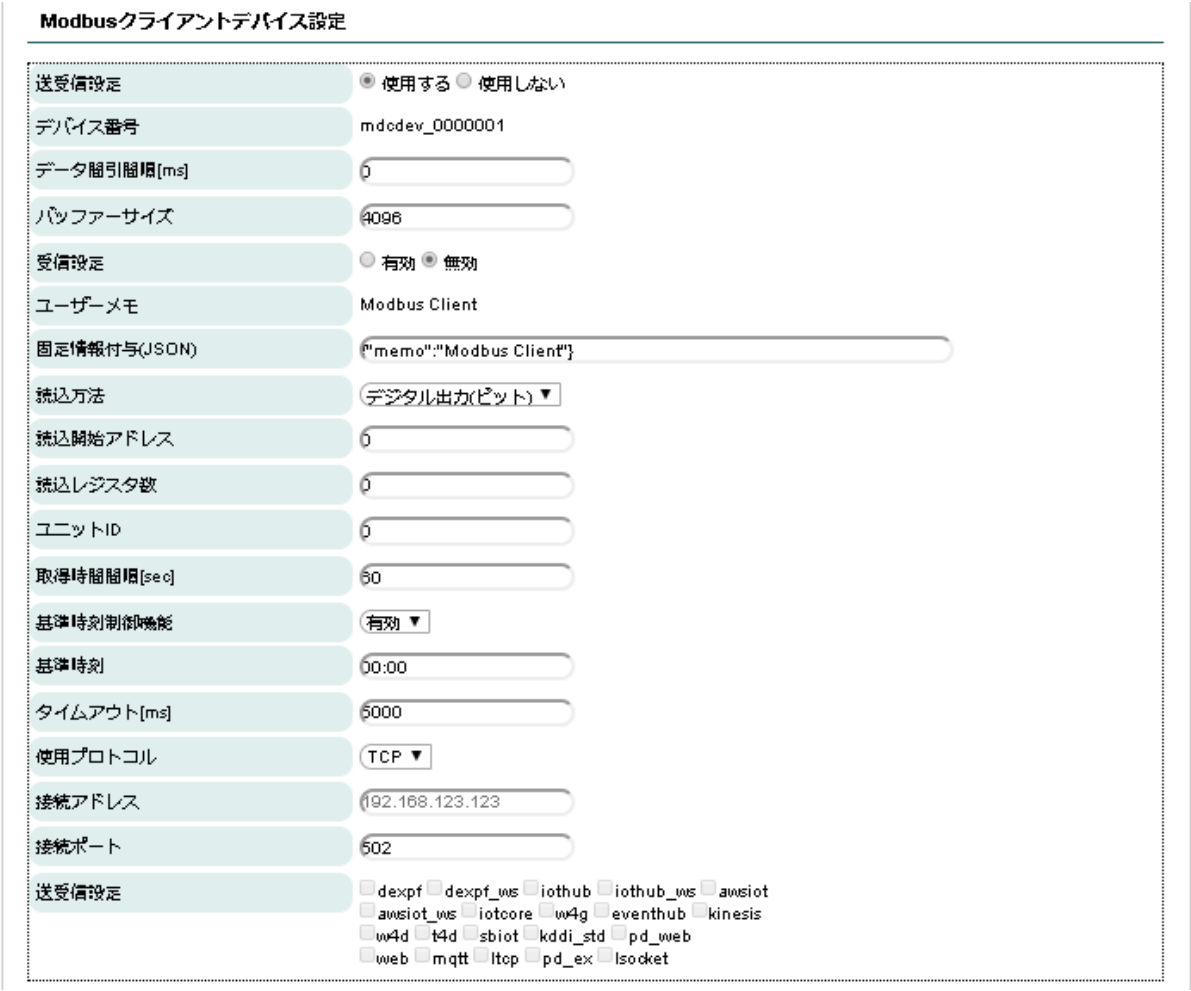


図 1: 「使用プロトコル」として「TCP」（ネットワーク）を選択した場合の表示

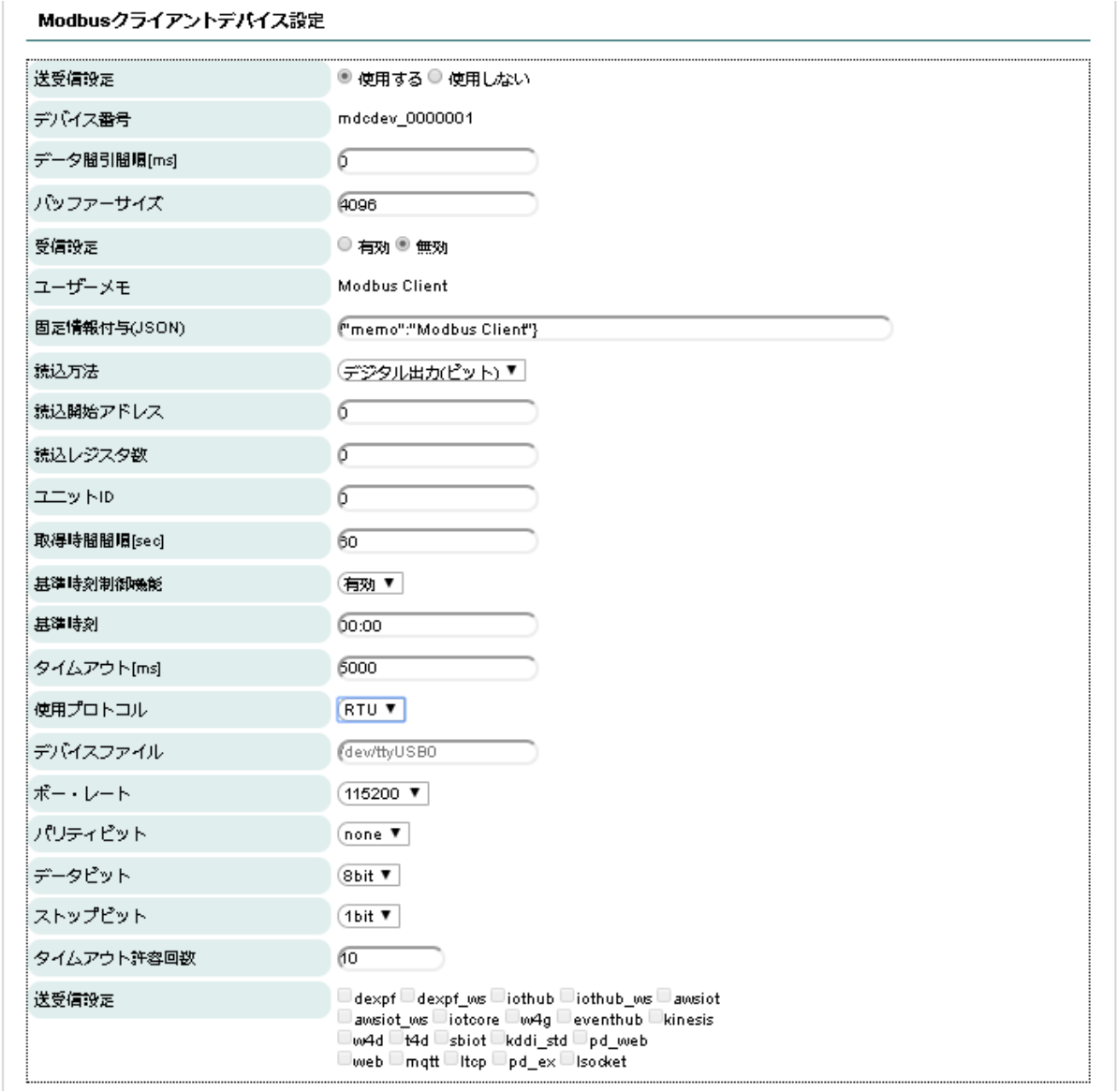


図 2: 「使用プロトコル」として「RTU」（シリアル）を選択した場合の表示

表 51: Modbus クライアントデバイス設定の設定項目 (1/2)

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
受信設定	PD Repeater を介してクラウドからの制御メッセージを受ける下流方向制御を行うか否かを選択します。
ユーザーメモ	登録されたデバイスにて設定されたメモ情報を表示します。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
読込方法	デジタル出力（ビット）、デジタル入力（ビット）、レジスタ出力、レジスタ入力から選択します。「デジタル出力」または「デジタル入力」を選んだ場合は、0 または 1 の並びが出力されます。
データタイプ	読込方法を「レジスタ出力」、「入力レジスタ」を選択した際に、出力のデータタイプを以下から選択します。 • 符号なし 16 ビット整数 • 符号付き 16 ビット整数 • 符号なし 32 ビット整数/リトルエンディアン • 符号付き 32 ビット整数/リトルエンディアン • 符号なし 32 ビット整数/ビッグエンディアン • 符号付き 32 ビット整数/ビッグエンディアン
読込開始アドレス	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。
読込レジスタ数	「読込方法」として「デジタル出力」または「デジタル入力」が選択されている場合は、読み込まれるビット数と解釈されます。　「開始アドレス」に設定されるアドレスから読み込むレジスタ数もしくビット数を設定します。
ユニット ID	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 の数値です。

表 52: Modbus クライアントデバイス設定の設定項目 (2/2)

設定項目	説明
取得時間間隔 [sec]	PLC デバイスからデータを取得する時間間隔を数字で設定します。単位は秒です。　後述の基準時刻制御を使用する場合、時間間隔は以下の値へと内部的に変更されます。 • 86400[sec] ×整数倍 • 43200[sec] • 28800[sec] • 21600[sec] • 14400[sec] • 10800[sec] • 7200[sec] • 3600[sec] • 1800[sec] • 900[sec] • 600[sec] • 300[sec]
基準時刻制御	特定の時刻にデータを取得する場合、本機能を有効とし基準時刻を設定しください。
基準時刻	特定の時刻にデータを取得する際の基準時刻を設定します。HH:MM 形式となります。
タイムアウト [ms]	PLC 機器からデータを取得する際のタイムアウトを設定します。単位はミリ秒です。
使用プロトコル	「TCP」、「RTU」のいずれかを選択します。「TCP」はネットワーク、「 RTU」はシリアルです。
接続アドレス	接続する PLC 機器の IP アドレスを設定します。
接続ポート	接続する PLC 機器の TCP ポート番号を設定します。　デフォルト値は、502 です。
読込デバイスファイル	PLC 機器を接続するシリアルポートのデバイスファイル名を設定します。
ボー・レート	シリアルポートのボー・レートを選択します。
パリティビット	シリアルポートのパリティビットを選択します。
データビット	シリアルポートのデータビット数を選択します。
ストップビット	シリアルポートのストップビット数を選択します。
タイムアウト許容回数	設定された回数を超えて、タイムアウトが連続して場合にシリアルポートをリセットします。　デフォルト値は、10 です。
送信先設定	“ 使用する ”を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

■ CSV ファイルを用いた取得 Modbus クライアントデバイスの拡張

WEB UI の「システム」→「ファイル管理」タブより、/var/webui/upload_dir にファイル名 pd_handler_modbus_client.csv の CSV ファイルを置くことで、WebUI 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
```

表 53: PD Handler Modbus Client の CSV ファイルの書式

パラメータ	データの形式	説明
デバイス番号	半角英数字	WebUI により割り振れたデバイス番号を記載します。 WebUI に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。

注釈:

- 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。

記載例

```
#localname,unit_id,read_function,data_type,read_addr,read_registers
mdcdev_0000001,15,read_coils,uint16_t,0x130,37
mdcdev_0000001,15,read_discrete_input,uint16_t,0x1c4,22
mdcdev_0000001,15,read_holding_registers,uint16_,0x160,3
mdcdev_0000001,16,read_input_registerss,uint32lsb_t,0x108,1
mdcdev_0000002,17,read_coils,uint16_t,0x130,37
mdcdev_0000002,18, read_discrete_input,u_int16,0x1c4,22
mdcdev_0000002,19,read_holding_registers,int16,0x160,3
mdcdev_0000002,20,read_input_registers,int32lsb,0x108,1
mdcdev_0000003,30,read_coils,u_int16,0x130,37
mdcdev_0000003,30,read_discrete_input,u_int16,0x1c4,22
mdcdev_0000003,31,read_holding_registers,u_int16,0x160,3
mdcdev_0000003,31read_,input_registers,u_int32msb,0x108,1
mdcdev_0000004,32,read_coils,u_int16,0x130,37
mdcdev_0000004,32,read_discrete_input,u_int16,0x1c4,22
mdcdev_0000004,33,read_holding_registers,int16,0x160,3
mdcdev_0000004,33,read_input_registers,int32msb,0x108,1
```

注釈:

- CSV ファイルが読み込まれると WebUI で設定された 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 WebUI で設定した 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。

■ 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。

「Modbus クライアントデバイス設定」メニューの「基準時刻制御」を「有効」にし、「取得時間間隔 [sec]」と「基準時刻」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。

表 54: 基準時刻制御における取得時間間隔の設定と実動作値

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」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。

20.1.7 Modbus サーバーデバイス設定

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。

表 55: PD Handler Modbus Server のレジスタマップ

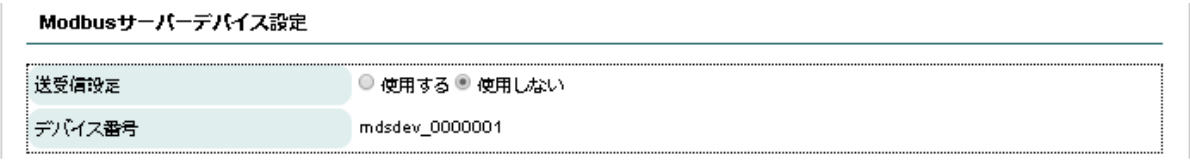
レジスタ	開始アドレス	サイズ
デジタル出力 (Coils)	0x000	uint8_t × 2048
デジタル入力 (Discrete Input)	0x000	uint8_t × 2048
レジスタ出力 (Holdig Registers)	0x000	uint16_t × 2048
レジスタ入力 (Input Registers)	0x000	uint16_t × 2048

WEB UI の「IoT データ」→「送受信設定」タブの「Modbus サーバーデバイス設定」メニューより、Modbus サーバーデバイスの設定を行うことが出来ます。

Modbus サーバーデバイスを利用するためには、WEB UI の「サービス」→「基本」→「Modbus(S) 登録」タブより、サービス機能の基本設定として Modbus サーバーデバイスが登録されている必要があります。

サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

「Modbus サーバーデバイス設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler MODBUS Server を「使用する」に設定して下さい。



登録済の Modbus サーバーデバイスが存在している場合、初期状態では上図のようになっています。(デバイスが 1 個登録されている場合です。)

デバイス毎に送信対象項目にて「送信する」を選択すると各設定項目が表示されます。

送信対象一括有効、送信対象一括無効ボタンにて全ての登録済のデバイスの送信対象を制御できます。

「送信する」を選択すると各設定項目が次のように表示されます。

■ Modbus サーバーデバイスの Device Type（待ち受けタイプ）が「TCP」で登録されている場合

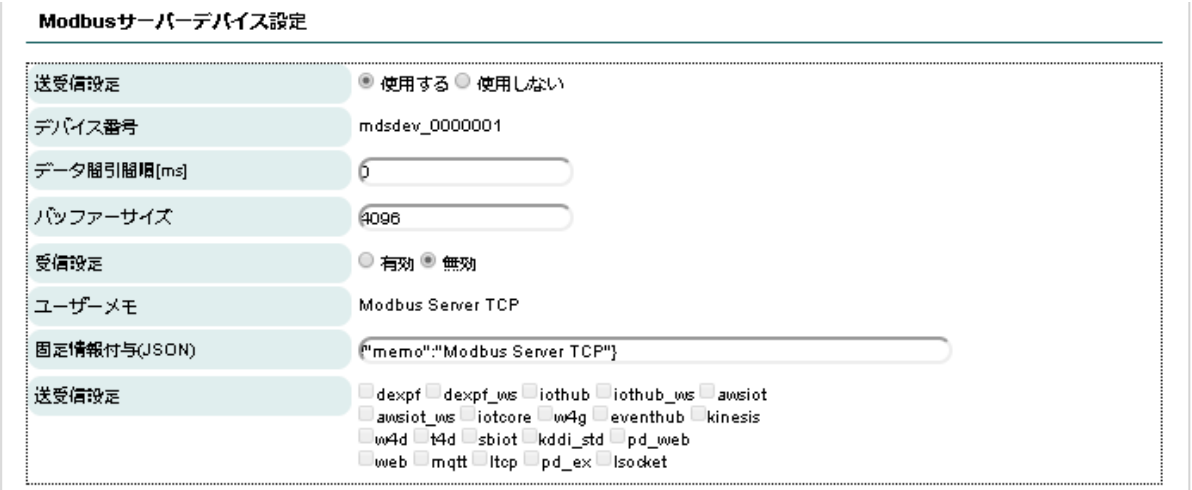


図 3: Device Type が「TCP」で登録されている場合の表示

表 56: Modbus サーバデバイス設定 (Device Type「TCP」) の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
受信設定	PD Repeater を介してクラウドからの制御メッセージを受けるか否かを選択します。
ユーザーメモ	登録されたデバイスにて設定されたメモ情報を表示します。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

注釈:

- TCP で待ち受ける場合、Modbus プロトコルの TCP ポート（502）を開放しておく必要があります。WEB UI の「システム」→「フィルター」タブの「フィルター開放設定」メニューにおいて「Modbus」を「有効」に設定して下さい。
- WEB UI から登録できる Device Type（待ち受けタイプ）が「TCP」の Modbus サーバーデバイスは、TCP ポート（502）の開放操作の兼ね合いから 1 つに制限されています。

■ Modbus サーバーデバイスの Device Type（待ち受けタイプ）が「RTU」で登録されている場合

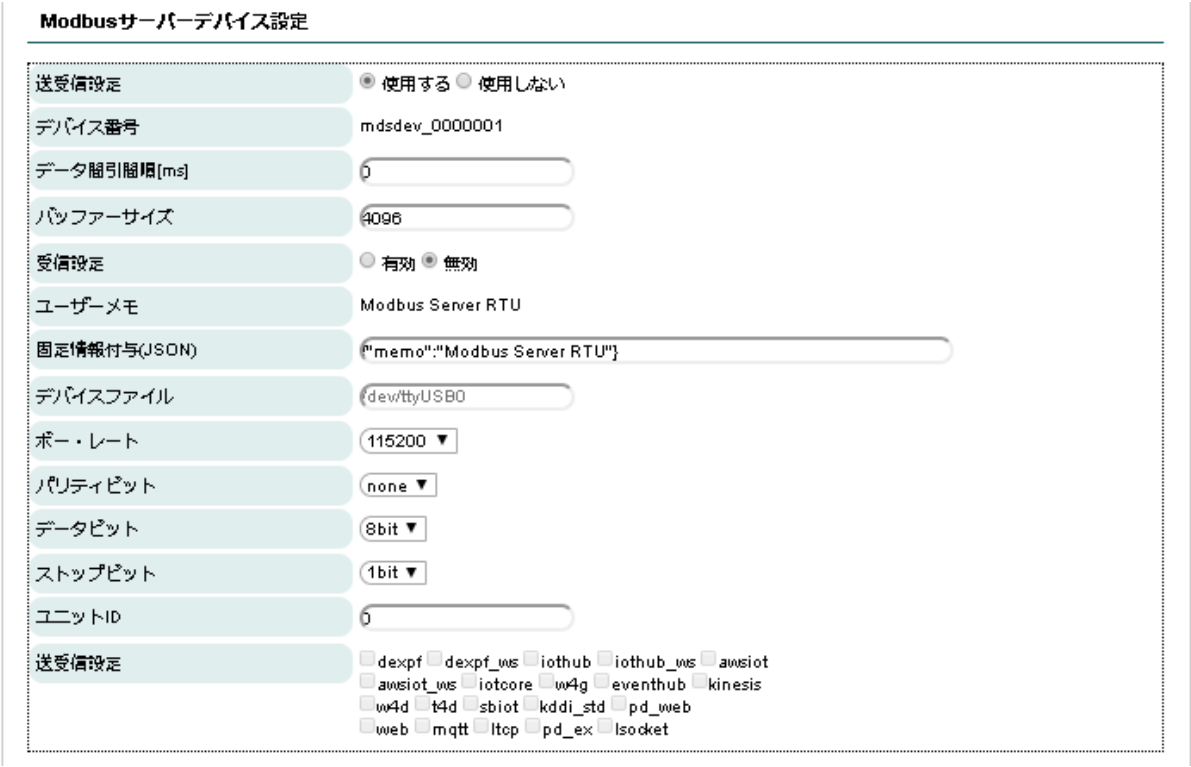


図 4: Device Type が「RTU」で登録されている場合の表示

表 57: Modbus サーバデバイス設定 (Device Type「RTU」) の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
受信設定	PD Repeater を介してクラウドからの制御メッセージを受けるか否かを選択します。
ユーザーメモ	登録されたデバイスにて設定されたメモ情報を表示します。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
デバイスファイル	PLC 機器からの接続を待ち受けるシリアルポートのデバイスファイル名を設定します。
ボー・レート	PLC 機器からの接続を待ち受けるシリアルポートのボー・レートを選択します。
パリティビット	シリアルポートのパリティビットを選択します。
データビット	シリアルポートのデータビット数を選択します。
ストップビット	シリアルポートのストップビット数を選択します。
ユニット ID	PLC 機器からの接続を待ち受ける Openbloks IoT Family 側の Modbus ユニット ID を設定します。ユニット ID は、１～247 の数値です。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

注釈:

- Device Type（待ち受けタイプ）が「RTU」の Modbus サーバーデバイスは、複数登録することが可能ですが、「デバイスファイル」の設定が重複した場合の動作は保障されません。
- Modbus サーバーデバイスが複数登録されている場合、レジスタマップは複数の Modbus サーバーデバイスで共有されることになります。Modbus サーバーデバイスとは Modbus サーバーの入出力デバイスと解釈して下さい。

20.1.8 Modbus2 クライアントデバイス設定

Modbus2 クライアントは、Modbus モデリングファイルに基づき Modbus プロトコルを用いて PLC 機器から定期的にデータを読み込み（ポーリングを行い）、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 Modbus モデリングファイルに基づく制御メッセージ (JSON 文字列) に基づき PLC 機器に Modbus プロトコルを用いて接続しデータを読み書きすることもできます。

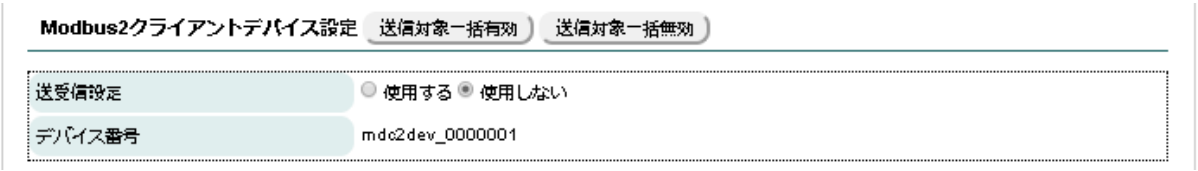
WEB UI の「IoT データ」→「送受信設定」タブの「Modbus2 クライアントデバイス設定」メニューより、Modbus2 クライアントデバイスの設定を行うことが出来ます。

Modbus2 クライアントデバイスを利用するためには、WEB UI の「サービス」→「基本」→「Modbus2(C) 登録」タブより、サービス機能の基本設定として Modbus2 クライアントデバイスが登録されている必要があります。

サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

「Modbus クライアントデバイス設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler MODBUS2 Client を「使用する」に設定して下さい。

Modbus モデリングファイルに PLC 機器のモデリング定義を追加したい場合は、**データハンドリング設定リファレンスガイド** の *PD Handler Modbus 2* を参照して下さい。



登録済の Modbus2 クライアントデバイスが存在している場合、初期状態では上図のようになっています。(デバイスが 1 個登録されている場合です。)

Modbus2 クライアントデバイスとは、PLC 機器そのものではなく、対象となる PLC 機器への接続形態（RTU プロトコルではデバイスファイル、TCP プロトコルでは IP アドレス）を意図します。

デバイス毎に送信対象項目にて「送信する」を選択すると各設定項目が表示されます。

送信対象一括有効、送信対象一括無効ボタンにて全ての登録済のデバイスの送信対象を制御できます。

「送信する」を選択すると各設定項目が次のように表示されます。

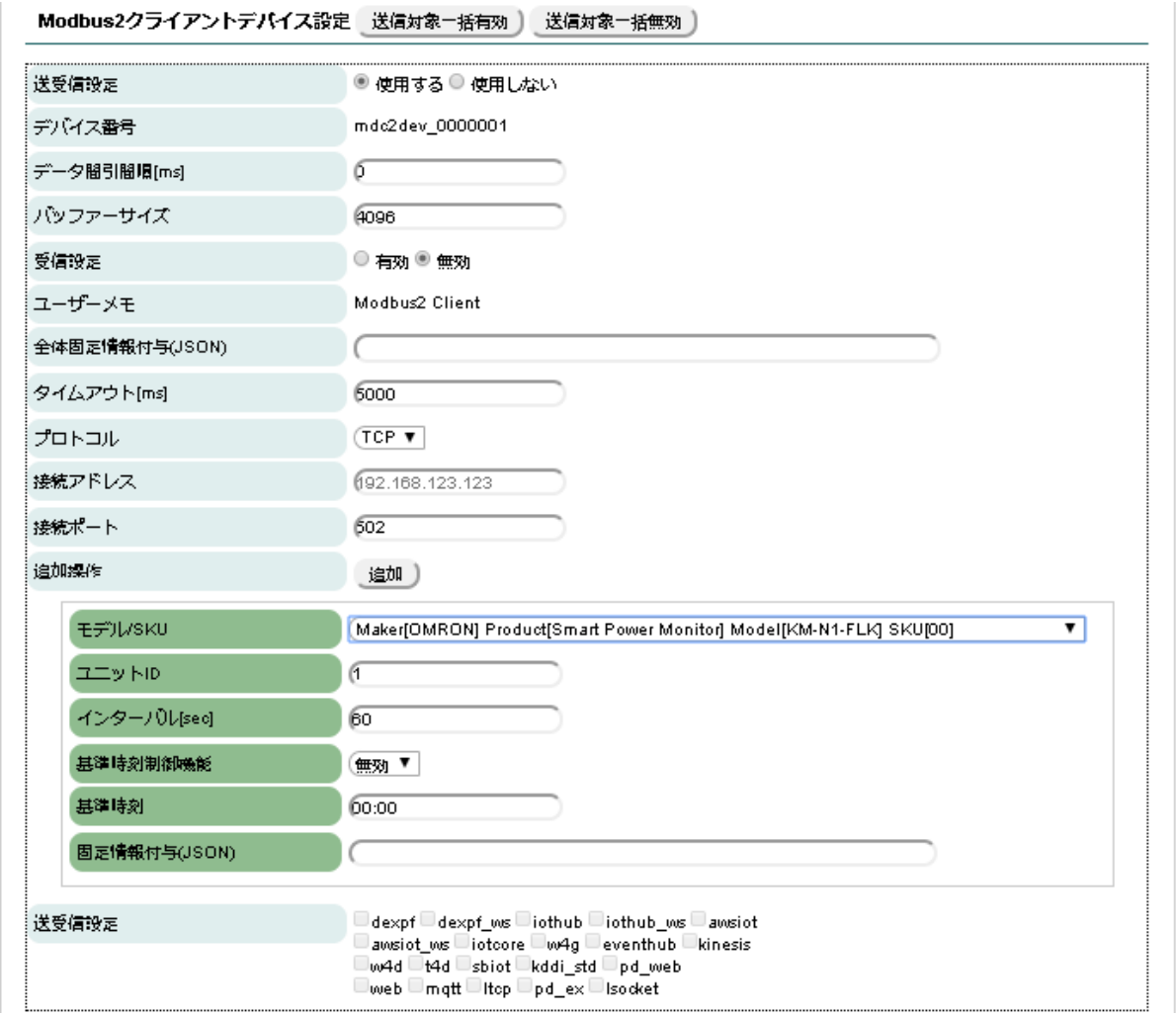


図 5: 「使用プロトコル」として「TCP」（ネットワーク）を選択した場合の表示

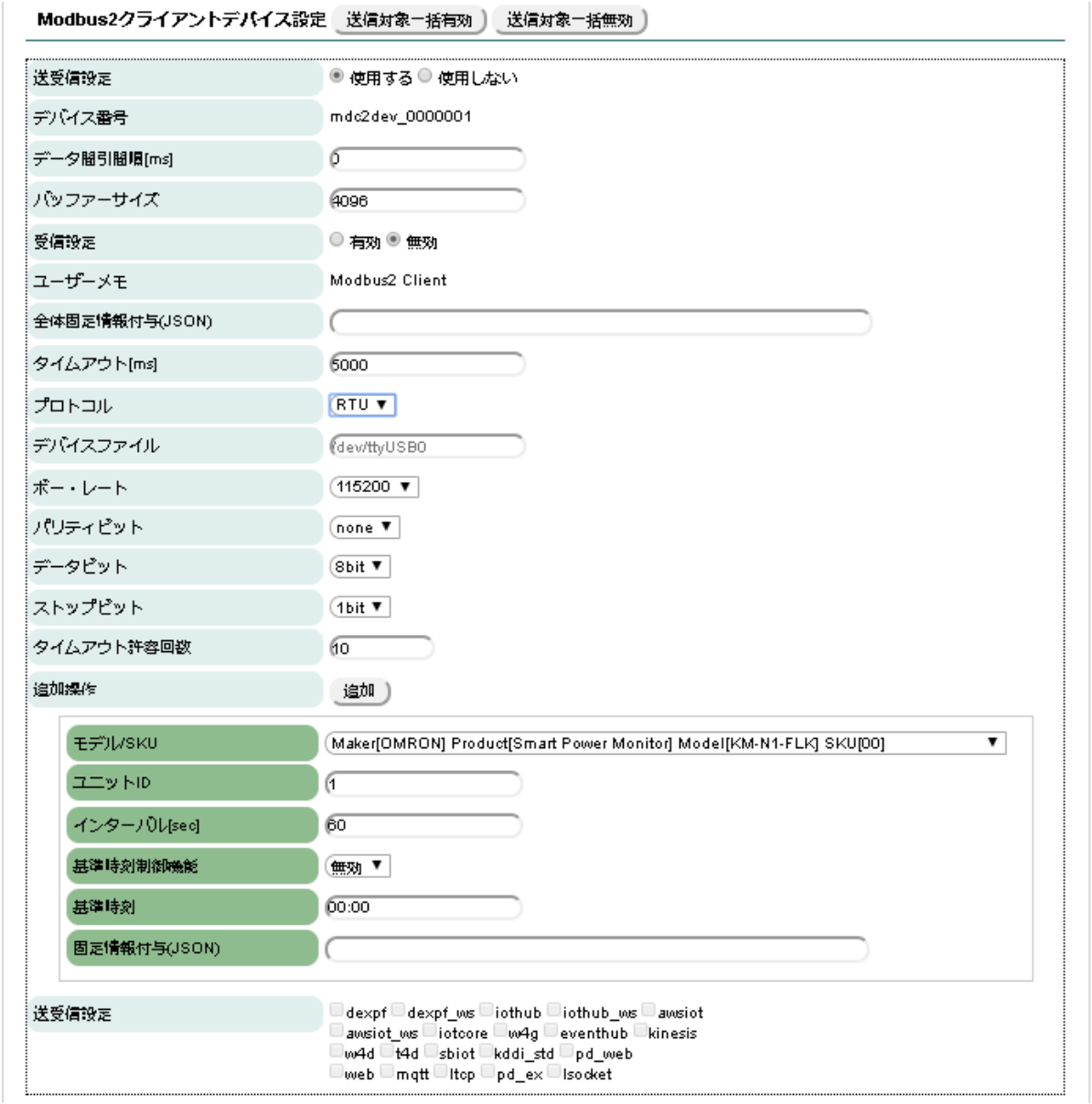


図 6: 「使用プロトコル」として「RTU」（シリアル）を選択した場合の表示

表 58: Modbus2 クライアントデバイス設定の設定項目 (1/2)

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
受信設定	PD Repeater を介してクラウドからの制御メッセージを受ける下流方向制御を行うか否かを選択します。
ユーザーメモ	登録されたデバイスにて設定されたメモ情報を表示します。
全体固定情報付与 (JSON)	データに付加する（個々の PLC 機器には依存しない）静的情報を JSON 文字列で設定します。
タイムアウト [ms]	PLC 機器からデータを取得する際のタイムアウトを設定します。単位はミリ秒です。
プロトコル	「TCP」、「RTU」のいずれかを選択します。「TCP」はネットワーク、「 RTU」はシリアルです。
接続アドレス	接続する PLC 機器の IP アドレスを設定します。
接続ポート	接続する PLC 機器の TCP ポート番号を設定します。　デフォルト値は、502 です。
デバイスファイル	PLC 機器を接続するシリアルポートのデバイスファイル名を設定します。
ボー・レート	シリアルポートのボー・レートを選択します。
パリティビット	シリアルポートのパリティビットを選択します。
データビット	シリアルポートのデータビット数を選択します。
ストップビット	シリアルポートのストップビット数を選択します。
タイムアウト許容回数	設定された回数を超えて、タイムアウトが連続して場合にシリアルポートをリセットします。　デフォルト値は、10 です。
追加操作	複数の PLC 機器からデータを取得する場合は、「追加」をクリックします。
モデル/SKU	プルダウンメニューから対象となる PLC 機器を選択します。表示される PLC 機器は Modbus モデリングファイルに記載されているものとなります。
ユニット ID:	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 の数値です。

表 59: Modbus2 クライアントデバイス設定の設定項目 (2/2)

設定項目	説明
インターバール [sec]	PL 機器からデータを取得する時間間隔を数字で設定します。　単位は秒です。後述の基準時刻制御を使用する場合、時間間隔は以下の値へと内部的に変更されます。 • 86400[sec] ×整数倍 • 43200[sec] • 28800[sec] • 21600[sec] • 14400[sec] • 10800[sec] • 7200[sec] • 3600[sec] • 1800[sec] • 900[sec] • 600[sec] • 300[sec] • 180[sec] • 120[sec] • 60[sec]
基準時刻制御	特定の時刻にデータを取得する場合、本機能を有効とし基準時刻を設定しください。
基準時刻	特定の時刻にデータを取得する際の基準時刻を設定します。HH:MM 形式となります。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

■ CSV ファイルを用いた取得 Modbus2 クライアントデバイスの拡張

WEB UI の「システム」→「ファイル管理」タブより、/var/webui/upload_dir にファイル名 pd_handler_modbus_2_client.csv の CSV ファイルを置くことで、WebUI で設定された１つのデバイス番号に対し複数の取得 Modbus2 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，ユニット ID 付加，メーカー名，プロダクト名，モデル名，SKU 名，
ユーザー定義固定キー１，ユーザー定義固定値１，ユーザー定義固定キー２，ユーザー定義固定値２，
インターバル、基準時刻制御、基準時刻
```

表 60: PD Handler Modbus2 Client の CSV ファイルの書式

パラメータ	データの形式	説明
デバイス番号	半角英数字	WebUI により割り振れたデバイス番号を記載します。WebUI に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。
ユニット ID 付加	半角英数字	ユニット ID を出力メッセージに付加するか否か、’0 ’又は ’１ ’, ‘ true ’, ’false ’
メーカー名	半角英数字	Modbus モデリングファイル記載されている PLC 機器のメーカー名を記載します。
プロダクト名	半角英数字	Modbus モデリングファイル記載されている PLC 機器のプロダクト名を記載します。
モデル名	半角英数字	Modbus モデリングファイル記載されている PLC 機器のモデル名を記載します。
SKU 名	半角英数字	Modbus モデリングファイル記載されている PLC 機器の SKU 名を記載します。
ユーザー定義固定キー１	半角英数字	データに付加する PLC 機器毎の静的情報のキーを記載します。
ユーザー定義固定値１	半角英数字	データに付加する PLC 機器毎の静的情報の値を記載します。
ユーザー定義固定キー２	半角英数字	データに付加する PLC 機器毎の静的情報のキーを記載します。
ユーザー定義固定値２	半角英数字	データに付加する PLC 機器毎の静的情報の値を記載します。
インターバル	半角数字	PL 機器からデータを取得する時間間隔を数字で設定します。単位は秒です。
基準時刻制御	半角英数字	基準時刻制御を行うか否か、’0 ’又は ’１ ’, ‘ true ’, ’false ’
基準時刻	半角英数字	特定の時刻にデータを取得する際の基準時刻を設定します。HH:MM 形式となります。

注釈:

- 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。
- 文字列にスペース (0x20) が含まれる場合は、"(0x22) 又は '(0x27) で囲んで下さい。

記載例

```
#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
mdc2dev_0000001,2,1,"Toukou Toushiba","SmaMe Series",S2MS-RNS22,0001,IBS,1-1,Mater,
↪2,300,true,00:00
mdc2dev_0000001,3,1,"Toukou Toushiba","SmaMe Series",S2MS-RNS22,0001,IBS,1-2,Mater,
↪1,300,true,00:00
mdc2dev_0000001,4,1,"Toukou Toushiba","SmaMe Series",S2MS-RNS22,0001,IBS,1-2,Mater,
↪2,300,true,00:00
```

注釈:

- CSV ファイルが読み込まれると WebUI で設定された Modbus2 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 WebUI で設定した Modbus2 クライアントデバイスを含む取得したい全ての Modbus2 クライアントデバイスを記載して下さい。

■ 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。

「Modbus2 クライアントデバイス設定」メニューの「基準時刻制御」を「有効」にし、「インターバル」と「基準時刻」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「インターバル」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。「インターバル」としてこれら以外の値が設定されると PD Handler Modbus2 Client 内で次のように扱われます。

表 61: 基準時刻制御における取得時間間隔の設定と実動作値

取得時間間隔の設定値	実動作値
0 〜 199	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」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。

20.1.9 Modbus2 サーバデバイス設定

Modbus2 サーバーは Modbus モデリングファイルに基づく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 Modbus モデリングファイルに基づき 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 Modbus モデリングファイルに基づく制御メッセージ (JSON 文字列) によりレジスタマップを読み書きすることもできます。

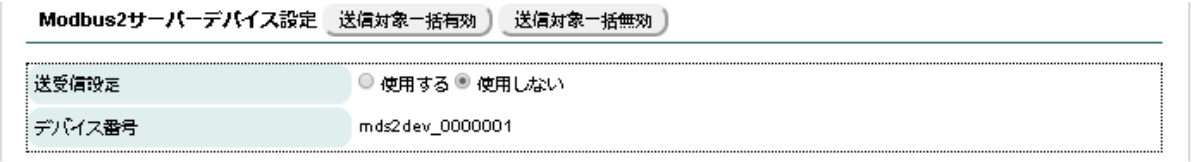
WEB UI の「IoT データ」→「送受信設定」タブの「Modbus2 サーバーデバイス設定」メニューより、Modbus サーバーデバイスの設定を行うことが出来ます。

Modbus2 サーバーデバイスを利用するためには、WEB UI の「サービス」→「基本」→「Modbus2(S) 登録」タブより、サービス機能の基本設定として Modbus2 サーバーデバイスが登録されている必要があります。

サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

「Modbus2 サーバーデバイス設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler MODBUS2 Server を「使用する」に設定して下さい。

Modbus モデリングファイルに PLC 機器のモデリング定義を追加したい場合は、**データハンドリング設定リファレンスガイド** の *PD Handler Modbus 2* を参照して下さい。



登録済の Modbus2 サーバーデバイスが存在している場合、初期状態では上図のようになっています。(デバイスが 1 個登録されている場合です。)

デバイス毎に送信対象項目にて「送信する」を選択すると各設定項目が表示されます。

送信対象一括有効、送信対象一括無効ボタンにて全ての登録済のデバイスの送信対象を制御できます。

「送信する」を選択すると各設定項目が次のように表示されます。

■ Modbus2 サーバーデバイスの Device Type（待ち受けタイプ）が「TCP」で登録されている場合

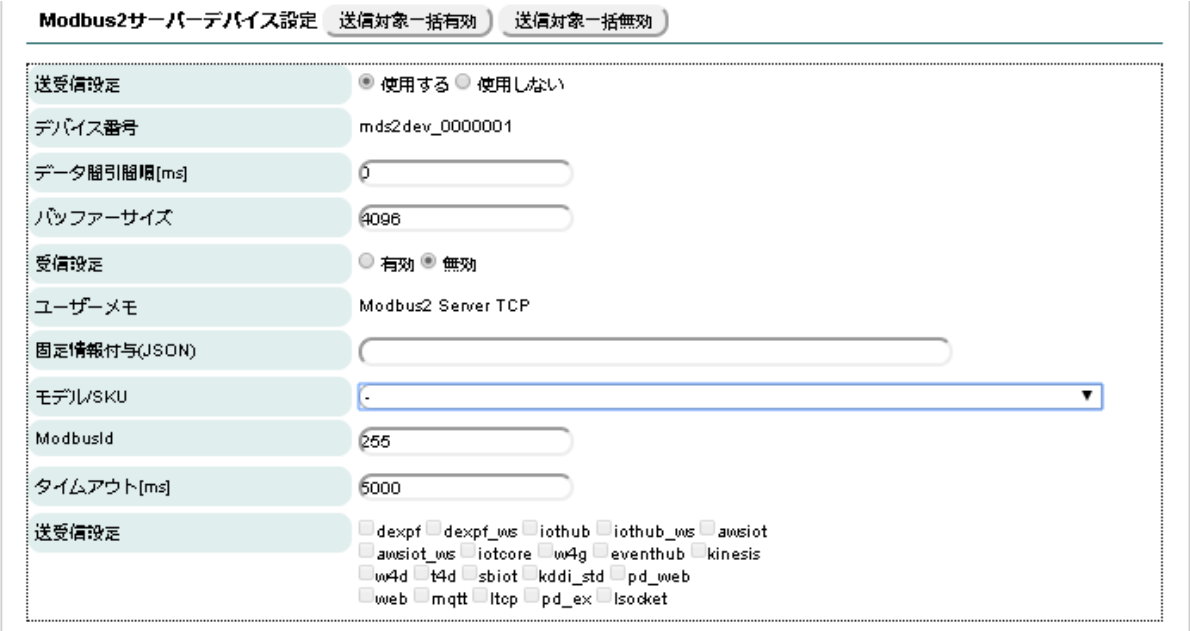


図 7: Device Type が「TCP」で登録されている場合の表示

表 62: Modbus2 サーバデバイス設定 (Device Type「TCP」) の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
受信設定	PD Repeater を介してクラウドからの制御メッセージを受けるか否かを選択します。
ユーザーメモ	登録されたデバイスにて設定されたメモ情報を表示します。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
モデル/SKU	プルダウンメニューからサーバーに使用するモデルを選択します。表示されるモデルは Modbus モデリングファイルに記載されているものとなります。
ユニット ID	サーバーの Modbus ユニット ID を設定します。ユニット ID は、1〜247 または 255 の数値です。
タイムアウト [ms]	PLC 機器からアクセスされる際のタイムアウトを設定します。単位はミリ秒です。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

注釈:

- TCP で待ち受ける場合、Modbus プロトコルの TCP ポート（502）を開放しておく必要があります。WEB UI の「システム」→「フィルター」タブの「フィルター開放設定」メニューにおいて「Modbus」を「有効」に設定して下さい。
- WEB UI から登録できる Device Type（待ち受けタイプ）が「TCP」の Modbus サーバーデバイスは、TCP ポート（502）の開放操作の兼ね合いから 1 つに制限されています。

■ Modbus2 サーバーデバイスの Device Type（待ち受けタイプ）が「RTU」で登録されている場合

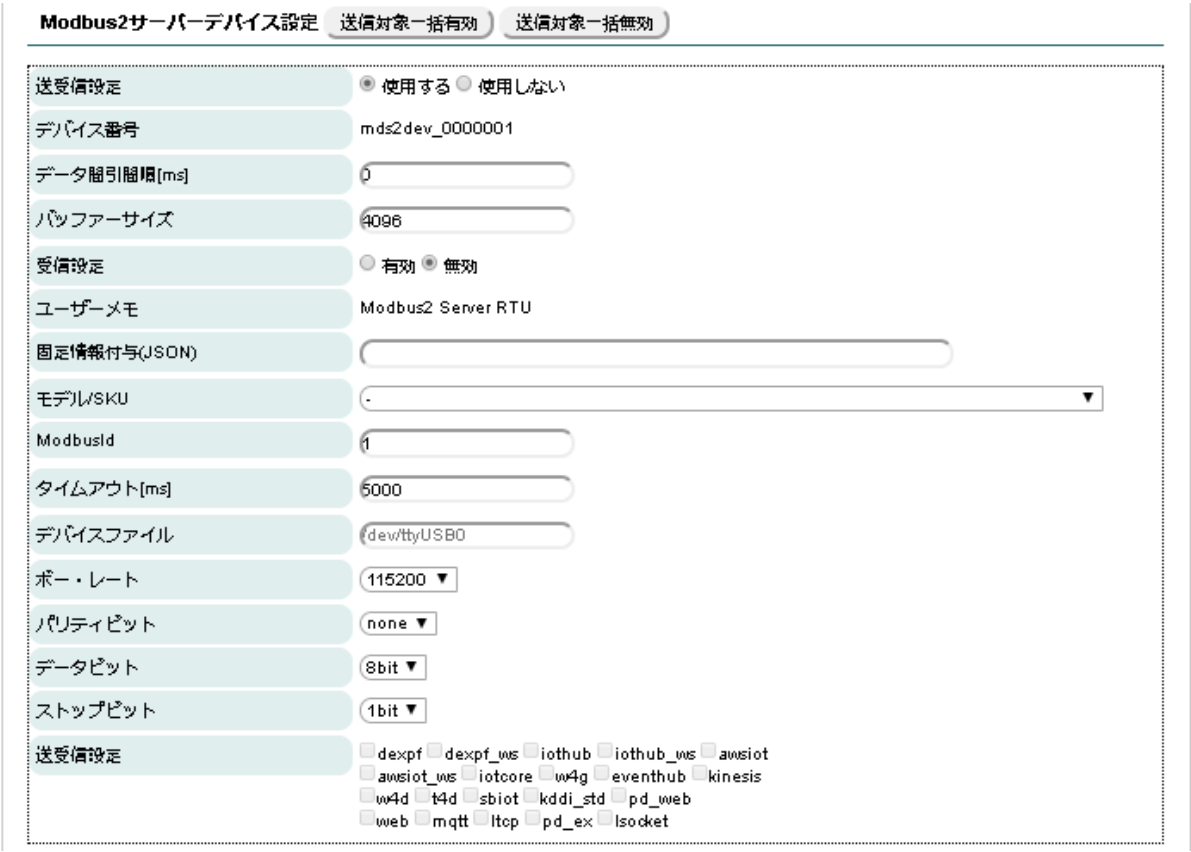


図 8: Device Type が「RTU」で登録されている場合の表示

表 63: Modbus2 サーバデバイス設定 (Device Type「RTU」) の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
受信設定	PD Repeater を介してクラウドからの制御メッセージを受けるか否かを選択します。
ユーザーメモ	登録されたデバイスにて設定されたメモ情報を表示します。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
モデル/SKU	プルダウンメニューからサーバーに使用するモデルを選択します。表示されるモデルは Modbus モデリングファイルに記載されているものとなります。
ユニット ID	サーバーの Modbus ユニット ID を設定します。ユニット ID は、1〜247 または 255 の数値です。
タイムアウト [ms]	PLC 機器からアクセスされる際のタイムアウトを設定します。単位はミリ秒です。
デバイスファイル	PLC 機器からの接続を待ち受けるシリアルポートのデバイスファイル名を設定します。
ボー・レート	PLC 機器からの接続を待ち受けるシリアルポートのボー・レートを選択します。
パリティビット	シリアルポートのパリティビットを選択します。
データビット	シリアルポートのデータビット数を選択します。
ストップビット	シリアルポートのストップビット数を選択します。
ユニット ID	PLC 機器からの接続を待ち受ける Openbloks IoT Family 側の Modbus ユニット ID を設定します。ユニット ID は、１〜247 の数値です。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

注釈:

- Device Type（待ち受けタイプ）が「RTU」の Modbus2 サーバーデバイスは、複数登録することが可能ですが、「デバイスファイル」の設定が重複した場合の動作は保障されません。

20.1.10 SW4x デバイス設定

PD Handler SW4x は、セイコーインスツル社製無線センサーネットワーク「ミスター省エネ」のベースノードとして機能し、無線センサーネットワークに接続される各種センサーノードからデータを取得し、PD Repeater を介してクラウドに送ります。

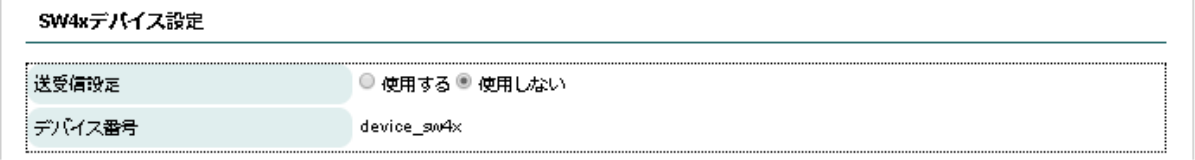
また、PD Repeater を介してクラウドから送られる制御メッセージ (JSON 文字列) に基づき Modbus RTU ノードに接続された PLC 機器の制御を行うこともできます。

WEB UI の「IoT データ」→「送受信設定」タブの「SW4x デバイス設定」メニューより、SW4x デバイスの設定を行うことが出来ます。

「SW4x デバイス設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler SW4x を「使用する」に設定して下さい。

Modbus RTU ノードを介した PLC 機器からのデータ取得は、PD Handler Modbus2 と同様に Modbus モデリングファイルに基づき行われます。

Modbus モデリングファイルに PLC 機器のモデリング定義を追加したい場合は、**データハンドリング設定リファレンスガイド** の *PD Handler SW4x* を参照して下さい。



初期状態では上図のようになっています。

「送信する」を選択すると各設定項目が下図のように表示されます。

表 64: SW4x デバイスの設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
受信設定	PD Repeater を介してクラウドからの制御メッセージを受けるか否かを選択します。
生データモード	取得したデータを変換せずに PD Repeater へ送ります。
デバイスファイル	ベースノード（SB4020 アドオンモジュール、SW4500 RS232 ベース又は SW4000USB ベース）を接続するシリアルポートのデバイスファイル名を設定します。 *1
無線グループ ID	無線センサーネットワークのグループ ID を設定します。SB4020 アドオンモジュール使用時のみ有効となります。
無線電力/チャンネル	無線センサーネットワークの無線電力とチャンネルを選択します。SB4020 アドオンモジュール使用時のみ有効となります。
RTT	通常は 500(msec) に設定します。Modbus RTU ノード使用時にタイムアウトが頻発する場合は、値を大きくします。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

注釈:

1. ベースノードを接続するシリアルポートの通信パラメータは、次の値に固定されています。SW4500 RS232 ベース又は SW4000USB ベースを用いる場合は、セイコーインスツル社が提供する設定用アプリケーションで設定して下さい。

   ボー・レート: 115200 bps

   パリティビット: none

   データビット: 8bit

   ストップビット: 1bit

■ Modbus RTU ノード

Modbus RTU ノードを介して PLC 機器からデータを取得する場合は、RTU モデリングファイル (/var/webui/config/pd_handler_sw4x_rtu.conf) を用意する必要があります。

RTU モデリングファイルの例を示します。

```
{
    "polls": [
       {
           "enable":  true,
           "maker": "Toukou Toushiba", "product": "SmaMe Series",
           "model": "S2MS-RNS22", "sku": "0001",
           "rid": "0xFB",
           "unitId": 1,
           "interval": 300, "time_sync": true, "base_time": "00:01"
       },
       {
           "enable":  true,
           "maker": "Toukou Toushiba", "product": "SmaMe Series",
           "model": "S2MS-RNS22", "sku": "0001",
           "rid": "0xFB",
           "unitId": 2,
           "interval": 300, "time_sync": true, "base_time": "00:01"
       }
    ]
}
```

RTU モデリングファイルの詳細については、**データハンドリング設定リファレンスガイド** の *PD Handler SW4x* を参照して下さい。

20.1.11 高圧スマートメータ (B ルート) 情報送信設定

PD Handler HVSMC を用いて高圧スマート電力量メータ (B ルート) から ECHONET Lite と AIF の仕様に基きデータを取得し、PD Repeater を介してクラウドに送ります。

WEB UI の「IoT データ」→「送受信設定」タブの「高圧スマートメータ (B ルート) 情報送信設定」メニューより、この設定を行うことが出来ます。

「高圧スマートメータ (B ルート) 情報送信設定」メニューが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Handler HVSMC を「使用する」に設定して下さい。

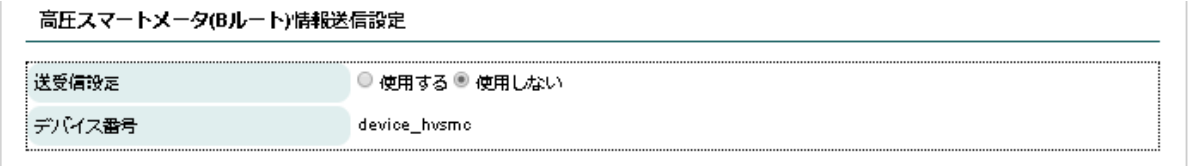
PD Handler HVSMC は高圧スマート電力量メーターから Ethernet 経由にて UDP ポート (3610) を用いてデータを待ち受けるため、同ポート開放しておく必要があります。

WEB UI の「システム」→「フィルター」タブの「フィルター開放設定」メニューにおいて「ECHONET(HVSMC)」を「有効」に設定して下さい。

また、使用する Ethernet インターフェースはネットワークタブにて有効にしてください。

尚、モバイル回線を使用する場合には、DHCP 設定は使用できませんのでダミーの IP アドレスを静的に設定する必要があります。

なお、Ethernet I/F を内蔵しない OpenBlocks シリーズ BX シリーズについては ECHONET Lite と AIF の認証が取得できないため、本機能はサポートされておりません。



初期状態では上図のようになっています。

「送信する」を選択すると各設定項目が下図のように表示されます。

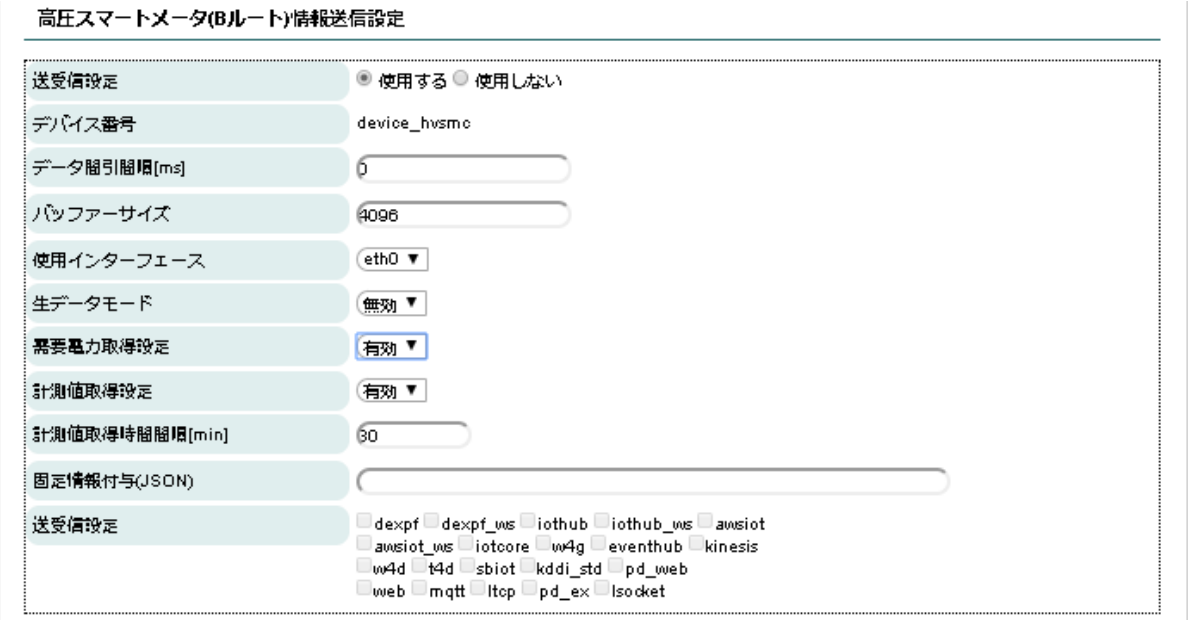


表 65: 高圧スマートメータ (B ルート) 情報送信設定の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
受信設定	PD Repeater を介してクラウドからの制御メッセージを受けるか否かを選択します。
使用インターフェース	高圧スマート電力量メータが接続される Ethernet I/F を選択します。
生データモード	取得したデータを変換せずに PD Repeater へ送ります。
需要電力取得設定	高圧スマート電力量メータから定時計測値が送られて来た際に最大需要電力の取得を行うか否かを設定します。
計測値取得設定	積算電力を以下の取得時間間隔で取得するか否かを設定します。
計測値取得時間間隔 [min]	積算電力の取得時間間隔を分単位で指定します。
固定情報付与 (JSON)	データに付加する静的情報を JSON 文字列で設定します。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

■ データの取得タイミング

PD Handler HVSMC は次の 4 つのタイミングでデータを取得し、PD Repeater に送ります。

1. PD Handler HVSMC の起動時
2. 高圧スマート電力量メータの自律動作による 30 分毎定時計測と需要電力取得
3. PD Handler HVSMC に設定された取得時間間隔に基づく計測値取得
4. 高圧スマート電力量メータからの動作状態通知時

以下に各取得タイミングにおける取得データとシーケンスを示します。
表中、EPC は ECHONET Propaty Code の略称です。

1. PD Handler HVSMC の起動時

表 66: PD Handler HVSMC の起動時に取得されるデータの一覧

データの種別	EPC	取得データ
動作状態	0x80	動作状態
ECHONET Lite 属性情報	0x82	規格バージョン番号
	0x9D	状態変移アナウンスプロパティマップ
	0x9E	Set プロパティマップ
	0x9F	Get プロパティマップ
高圧スマート電力量メータ属性情報	0x8D	製造番号
	0xD3	係数
	0xD4	係数の倍率
	0xE0	確定日
	0xE5	算有効電力量有効桁数
	0xE6	積算有効電力量単位
	0xC4	需要電力有効桁数
	0xC5	需要電力単位
	0xC7	累積最大需要電力単位
	0xCC	力測積算無効電力量 (遅れ) 有効桁数
	0xCD	力測積算無効電力量 (遅れ) 単位

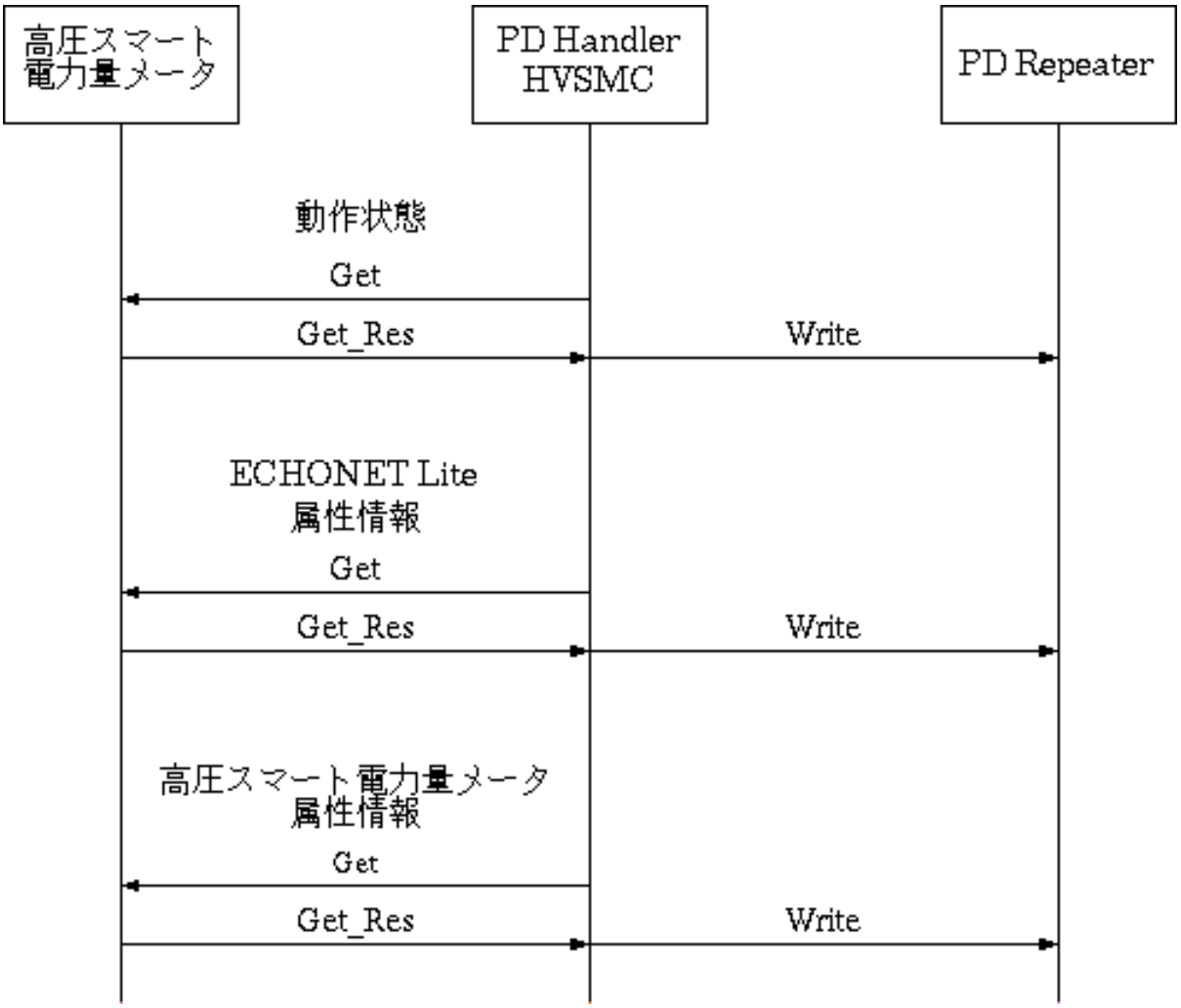


図 9: PD Handler HVSMC 起動時のデータ取得タイミング

2. 高圧スマート電力量メータの自律動作による 30 分毎の定時計測と需要電力取得

表 67: 定時計測と需要電力取得データ一覧

データの種別	EPC	取得データ
定時計測値 (毎時 0 分、30 分)	0xE3	定時積算有効電力量計測値
	0xC3	定時需要電力
	0xCB	定時力測積算無効電力量 (遅れ) 計測値
需要電力 (定時計測値を受信したタイミングでデータ要求)	0xC1	月間最大需要電力
	0xC2	累積最大需要電力

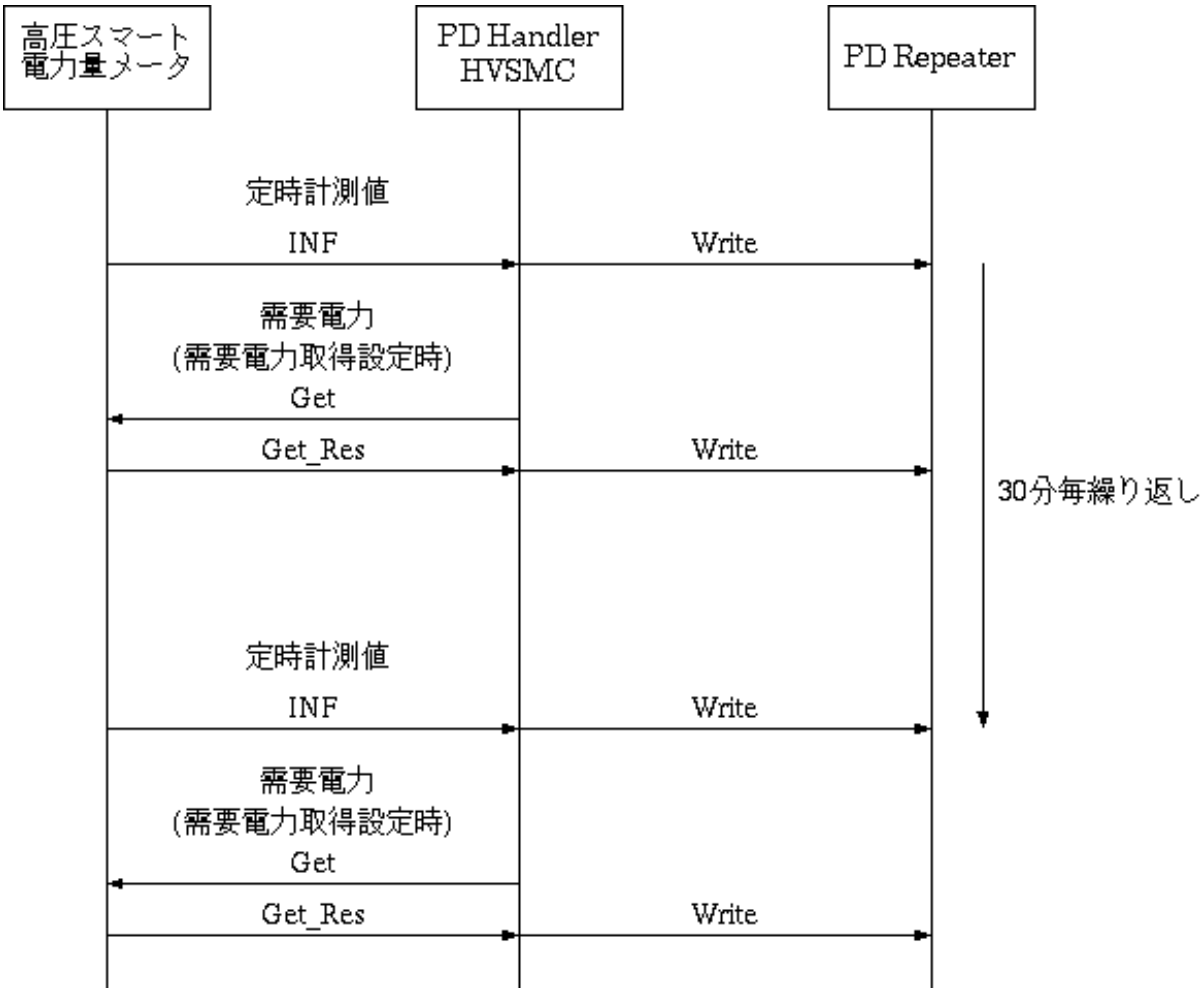


図 10: 需要電力取得のタイミング

3. PD Handler HVSMC に設定された取得時間間隔に基づく計測値取得

表 68: 計測値取得データ一覧

データの種別	EPC	取得データ
計測値	0xE2	積算有効電力量計測値
	0xE4	力測積算有効電力量計測値
	0xCA	力測積算無効電力量 (遅れ) 計測値

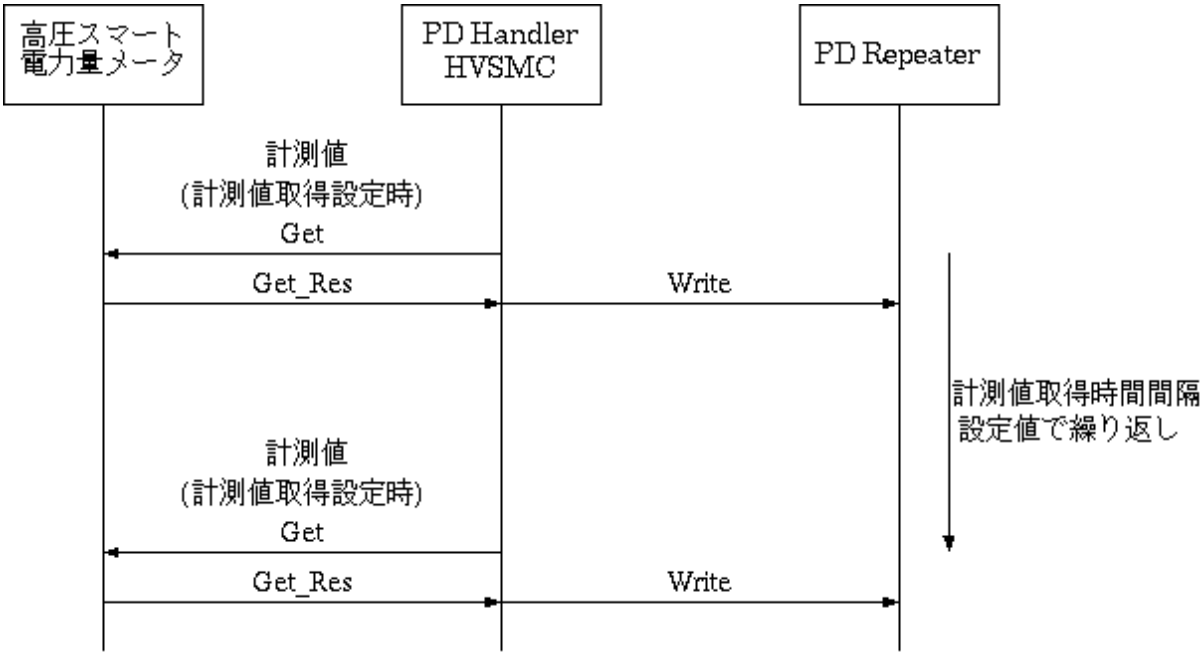


図 11: 計測値取得タイミング

4. 高圧スマート電力量メータからの動作状態通知時

表 69: 動作状態通知時の取得データ一覧

データの種別	EPC	取得データ
動作状態	0x80	動作状態
異常発生状態	0x88	異常発生状態

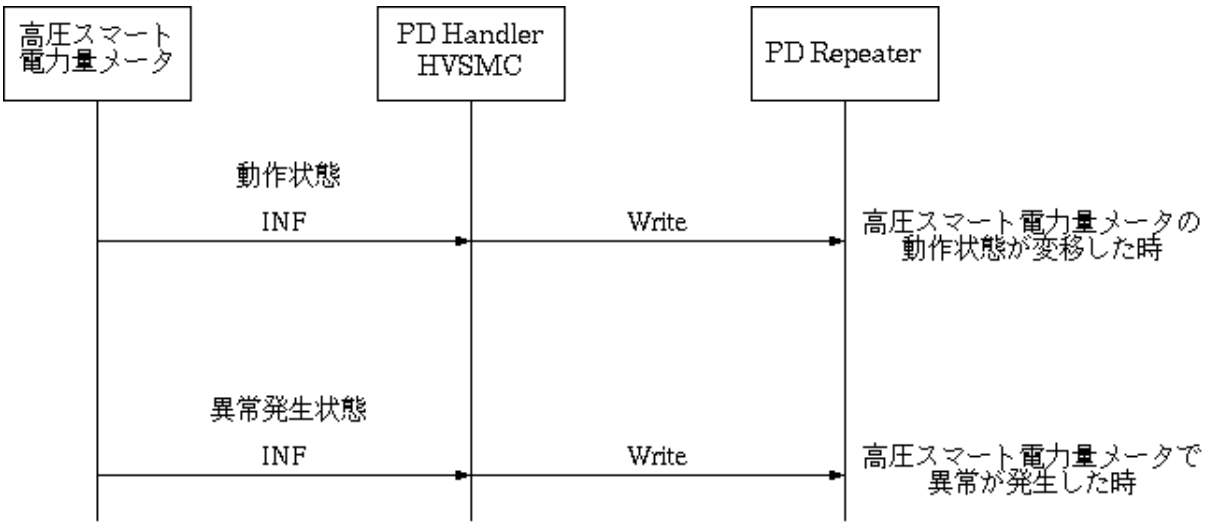


図 12: 動作状態通知のタイミング

20.1.12 デバイス設定 (ユーザー定義)

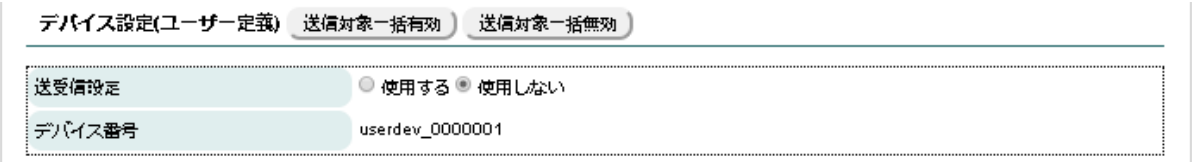
OpenBlocks シリーズの標準データ収集アプリケーション (PD Handler) を用いず、ユーザーが独自に用意するアプリケーションによりデータをハンドリングし、PD Repeater と連携させることができます。

WEB UI の「IoT データ」→「送受信設定」タブの「デバイス設定（ユーザー定義）」メニューより、ユーザー定義のデバイスの設定を行うことが出来ます。

ユーザー定義のデバイスを利用するためには、WEB UI の「サービス」→「基本」→「ユーザーデバイス登録」タブより、サービス機能の基本設定としてユーザー定義のデバイスが登録されている必要があります。

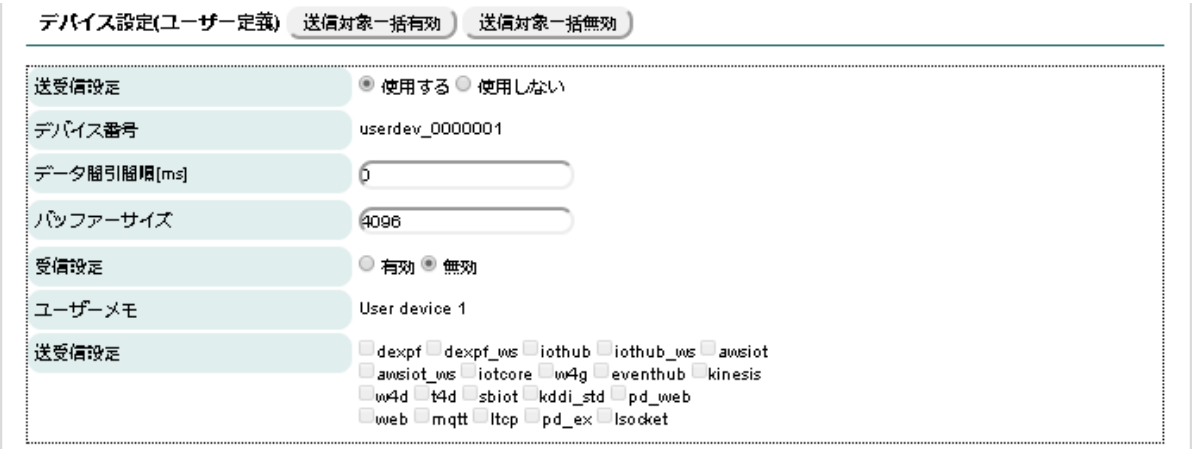
サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

ユーザー定義のデバイスの登録とは、デバイス番号を割り振りメモ情報を付与する作業です。



登録済のユーザー定義のデバイスが存在している場合、初期状態では上図のようになっています。(デバイスが 1 個登録されている場合です。)

デバイス毎に送信対象項目にて「送信する」を選択すると各設定項目が表示されます。
送信対象一括有効、送信対象一括無効ボタンにて全ての登録済のデバイスの送信対象を制御できます。



「送信する」を選択すると各設定項目が上図のように表示されます。

表 70: デバイス設定 (ユーザー定義) の設定項目

設定項目	説明
デバイス番号	OpenBlocks シリーズの WEB UI 内で管理している番号です。変更はできません。
データ間引間隔	データを間引くための入力データを受け取らない時間を msec 単位で設定します。0 の場合、間引きは行われません。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです。
ユーザーメモ	登録されたデバイスにて設定されたメモ情報を表示します。
送信先設定	"使用する"を選択した送信先に対してチェックボックスが選択できるようになります。チェックを付けた送信先に対して、送信を行います。チェックをつけると送信先固有の設定項目が表示されます。送信先固有の設定については、**送受信先毎の設定** を参照して下さい。

20.1.13 下流方向制御

クラウド等からメッセージを受け取り Modbus 対応 PLC 機器等を制御する下流方向の制御機能について説明します。

下流方向制御の概要

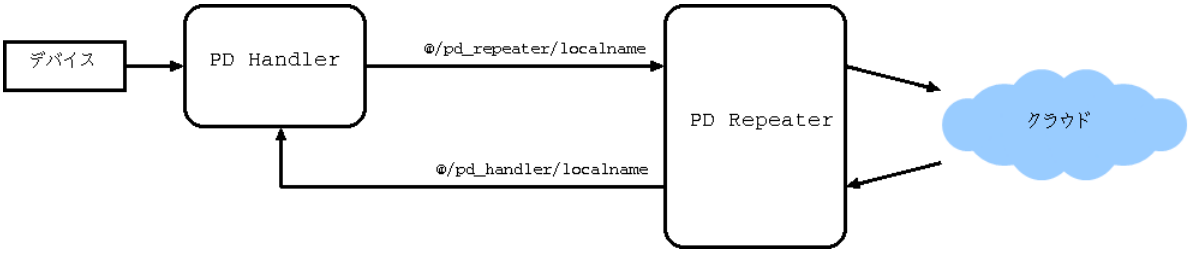
IoT データ制御機能を提供するソフトウェアモジュールにおいては、そのプロセス間通信において UNIX ドメインソケットを使用しています。下流方向制御も UNIX ドメインソケットを用いて動作します。

下流方向制御は PD Repeater が対応する送信先の内、MQTT プロトコルに対応したクラウドサービスと PD Exchange(pd_ex)、PH 社独自仕様 WEB サーバー (pd_web) 並び TCP(ltcp) 接続に対応します。

MQTT プロトコルと TCP 接続においては、送受 (パブ／サブ) 兼用の常時接続状態で動作し、PD Exchange については定期的に制御メッセージが無いかポーリングを行います。

PH 社独自仕様 WEB サーバーにおいては、データ送信時の応答メッセージを下流方向への制御メッセージとして扱います。

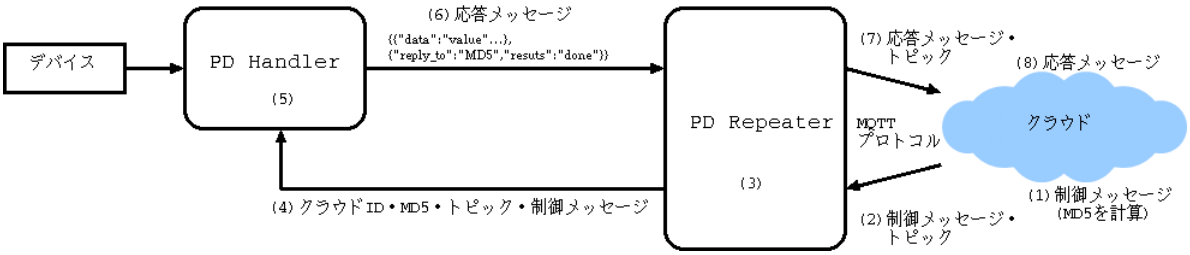
上流（クラウド）からの制御メッセージは PD Repeater を介し UNIX ドメインソケットにより PD Handler Modbus Client 等の下流のソフトウェアモジュールへ転送されます。下流のソフトウェアモジュールはデータの送信に用いる UNIX ドメインソケットにより PD Repeater を介しクラウドへ制御結果を返します。



制御メッセージは JSON 文字列を想定していますが、クラウド側の制御要件としてペイロードに載せられない情報、MQTT においてはトピック、PD Exchange においてはコマンド ID、PH 社独自仕様 WEB サーバーにおいては応答ヘッダー、TCP 接続においてはポート番号が必要となる場合が考えられるため、PD Repeater から下流のソフトウェアモジュールへの制御メッセージの転送においては、これらの情報と制御メッセージのハッシュ値 (MD5) を付加しています。

制御メッセージのハッシュ値は、下流のソフトウェアモジュールが制御結果を返す際に、どの制御メッセージに対する応答なのか明確にすることを想定したものです。

MQTT プロトコルで接続されるクラウドを例に下流方向制御の流れを説明します。



1. 制御メッセージのハッシュ値 (MD5) を計算しておきます。
2. MQTT プロトコルにより制御メッセージをパブリシュします。
3. 制御メッセージを受け、制御メッセージのハッシュ値 (MD5) を計算します。
4. クラウド ID（PD Repeater 上の便宜上の番号)、制御メッセージのハッシュ値 (MD5)、トピックと共に制御メッセーを下流のソフトウェアモジュールに転送します。
5. 制御メッセージに従って処理を行います。
6. 処理結果と共に制御メッセージのハッシュ値 (MD5) を応答メッセージに載せて返します。
7. MQTT プロトコルにより制御メッセージをパブリシュします。
8. 応答メッセージ上のハッシュ値 (MD5) と（１）で計算したハッシュ値 (MD5) を比較することで、どの制御メッセージに対する応答であるかを明確にすることができます。

PD Repeater の下流方向メッセージ

OpenBlocks シリーズに搭載される IoT データ制御機能をそのまま利用する上で本項目は特に意識する必要はありません。

■ PD Repeater が流のソフトウェア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。

表 71: メッセージフォーマット

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

表 72: メッセージフォーマットの説明

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

■ Cloud ID とヘッダーの内容を示します。

表 73: Cloud ID とヘッダーの内容

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		
Watson IoT for Device	0x02	対応	MQTT 受信トピック *1
Amazon Kinesis	0x03		
<予約>	0x04		
<予約>	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 *1
MS Azure IoT Hub	0x0a	対応	MQTT 受信トピック *2
Toami for DOCOMO	0x0b		
ドメインソケット	0x0c		
KDDI IoT クラウド Standard	0x0d		
<予約>	0x0e		
PH 社独自仕様 WEB サーバー	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 *3
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 *2
<予約>	0x14		
SoftBank スマ可視専用クラウド	0x15	対応	MQTT 受信トピックと oneM2M の JSON 制御用オブジェクト *4
<予約>	0x16		
<予約>	0x17		

注釈:

1. 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。
2. 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。
3. 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> }
```

4. SoftBank スマ可視専用クラウドについては、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。

Modbus クライアント／サーバーの下流方向制御

PD Handler Modbus の下流方向制御について説明します。

PD Handler Modbus の下流方向制御を利用するためには、Modbus に用いられるファンクションコードを理解しておく必要があります。

PD Handler Modbus に用いられているファンクションコードを示します。

表 74: PD Handler Modbus に用いられている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みます。
0x02	Read Discrete Input	デジタル入力のビット値を読み込みます。
0x03	Read Holding Registers	レジスタ出力に設定されている値を読み込みます。
0x04	Read Input Registers	レジスタ入力の値を読み込みます。
0x05	Write Single Coil	デジタル出力 1 ビットのビット値を設定します。
0x06	Write Single Register	レジスタ出力 1 レジスタの値を設定します。
0x07	Read Exception Status	クライアント／サーバー間でエラーステータスを通知します。
0x09	Write Single Discrete Input	デジタル入力 1 ビットのビット値を設定します。
0x0a	Write Single Input Register	レジスタ入力 1 レジスタの値を設定します。
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します。
0x10	Write Multiple Registers	連続する複数のレジスタ出力の値を設定します。
0x11	Report Slave ID	接続可能なスレーブ（サーバー）機器の ID を通知します。
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します。
0x14	Write Multiple Input Registers	連続する複数のレジスタ入力の値を設定します。
0x16	Mark Write Registers	レジスタ出力をマスクします。
0x17	Write and Read Registers	連続する複数のレジスタ出力の値を設定し、その値を読み込みます。

ここで、0x09, 0x0a, 0x13, 0x14 物理的な入力を持たない 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 プロトコルには存在しない独自拡張のファンクションです。

Modbus ファンクションコードは、PD Handler Modbus のコンフィグファイル (pd_handler_modbus_client.conf, pd_handler_modbus_server.conf) もしくは下流方向制御において JSON 文字列の"function"キーに設定されます。

また、CVS ファイル（pd_handler_modbus_client.csv）においては「読込方式」として第 3 カラムに設定されます。

"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"function"キー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。

"function" キーに用いるファンクションコードの別称を示します。

表 75: "function"キーに用いるファンクションコードの別称

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_input_register
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_register
0x11	Report Slave ID	17	report_slave_id
0x13	Write Multiple Discrete Input	19	write_muliple_discrete_input
0x14	Write Multiple Input Registers	20	write_multiple_input_registers
0x16	Mark Write Registers	22	mark_write_registers
0x17	Write and Read Registers	23	write_and_read_registers

Modbus クライアント

PD Handler Modbus Client の下流方向制御について説明します。

PD Handler Modbus Client の下流方向制御に用いる JSON 文字列のオブジェクトを示します。

表 76: PD Handler Modbus Client の下流方向制御に用いる JSON 文字列のオブジェクト

キー	データ型	必須	説明
protocol	文字列	○	TCP 接続機器の場合は ’"tcp"、RTU 接続機器の場合は "rtu"
node	文字列	△	TCP 接続 PLC 機器の IP アドレス
port	整数	△	TCP 接続 PLC 機器のポート番号
device	文字列	△	RTU 接続 PLC 機器のデバイスファイル
unit	整数	○	PLC 機器の ModbusID、1〜247 又は 255(TCP 接続のみ)
function	文字列又は整数	○	*Modbus* **クライアント／サーバーの下流方向制御** に説明するファンクションコードの内、次表 「PD Handler Modbus Client で利用可能なファンクションコード」に記載されるコード
data_type	文字列又は整数		次の何れか **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 ’ 指定されない場合は、符号なし 16 ビット整数
address	文字列又は整数		データが格納されている、あるいは格納する 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。指定されない場合は、0x00
number	文字列又は整数		読み書きするレジスタ数を記載します。指定されない場合は、1
values	整数配列	△	書き込むデータ（整数値）の並び

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。

表 77: PD Handler Modbus Client で利用可能なファンクションコード

コード	名称	利用可能
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	Mark Write Registers	
0x17	Write and Read Registers	○

例えば、TCP 接続されている入力レジスタを読み込むのであれば、クラウドより次のような制御メッセージを送ります。

```
{
  "protocol": "tcp", "node": "192.168.1.8", "port": 502, "unit": 255,
  "function": "0x04", "data_type": "uinit16_t", "address": "0x160", "number": 5
}
```

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、制御メッセージのハッシュ値（MD5）です。

シリアル接続の PLC 機器の出力レジスタに値を書き込むのであれば、クラウドより次のような制御メッセージを送ります。

```
{
  "protocol": "rtu", "device": "/dev/ttyEX1",
  "unit": 16, "function": "0x10", "data_type": "uint32lsb_t", "address": "0x120",
  "values": [4567, 891011]
}
```

ここで"data_type"が 32bits の場合、"values"は上位／下位の 16bits に分割されて処理されるため、"number"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。

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]
}
```

なお、"protocol","node","port","device"で指定される PLC 機器（への接続方法）は、Modbus クライアントデバイスとして登録され使用設定が有効になっているものに限られます。

Modbus サーバ

PD Handler Modbus Server の下流方向制御について説明します。

PD Handler Modbus Server の下流方向制御に用いる JSON 文字列のオブジェクトを示します。

表 78: PD Handler Modbus Server の下流方向制御に用いる JSON 文字列のオブジェクト

キー	データ型	必須	説明
function	文字列又は整数	○	*Modbus* **クライアント／サーバーの下流方向制御** に説明するファンクションコードの内、次表 「PD Handler Modbus Server で利用可能なファンクションコード」に記載されるコード
data_type	文字列又は整数		次の何れか **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 ’ 指定されない場合は、符号なし 16 ビット整数
address	文字列又は整数		データが格納されている、あるいは格納する 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。指定されない場合は、0x00
number	文字列又は整数		読み書きするレジスタ数を記載します。指定されない場合は、1
values	整数配列	△	書き込むデータ（整数値）の並び

PD Handler Modbus Client とは異なり PD Handler Modbus Server 自身のレジスタマップに対する操作を行うものであるため、"node"や"device"キーの設定は必要としません。

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。

表 79: PD Handler Modbus Server で利用可能なファンクションコード

コード	名称	利用可能
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	Mark Write Registers	
0x17	Write and Read Registers	○

例えば、入力レジスタを読み込むのであれば、クラウドより次のような制御メッセージを送ります。

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

}
```

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 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、制御メッセージのハッシュ値（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 からは異なるデバイス番号を持つ個別の Modbus サーバーデバイスと位置付けられるため、応答メッセージには、PD Repeater から制御メッセージを受けた 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。

入力レジスタに値を書き込むのであれば、クラウドより次のような制御メッセージを送ります。入力レジスタは本来物理的に AD コンバータ等を介してセットされるレジスタですが、PD Handler Modbus Server では下流方向制御によりレジスタマップを書き換えることができます。

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

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

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/ttyEX1",
  "unit":17, "address":320, "function":20, "data_type": "uint32_t",
  "values": [4567, 8910, 561, 435]
}
```

Modbus 2 クライアント／サーバーの下流方向制御

Modbus 2 クライアント

PD Handler Modbus2 Client の下流方向制御について説明します。

PD Handler Modbus2 は、PD Handler Modbus とは異なり Modbus モデリングファイルに基づきデータの読み書きを行うため、個々に Modbus ファンクションやレジスタのアドレスを指定する必要はありません。

例えば unitId 3 の PLC 機器のデータを取得するのであれば、その制御メッセージは次のように unitId を指定するだけで済みます。

```
{
  "unitId": 3
}
```

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、制御メッセージのハッシュ値（MD5）です。

ポーリン動作で取得されるデータに対し、下流方向制御で取得されるデータには Modbus モデリングファイルで disable 定義されているレジスタセットのデータが含まれます。

なお、特定のレジスタセットのデータのみを取得することはできません。

PLC 機器の出力レジスタ (outputBits または outputRegisters) に値を書き込む場合は、クラウドより次のような制御メッセージを送ります。

```
{
  "unitId": 3, "write":{"doValue_ch1": 1, "doValue_ch3": 1,"pulseCountReset_ch1":␣
↪1}
}
```

ここで、write キーに与えられるオブジェクトは、Modbus モデリングファイルの outputBits または inputRegisters に定義される、,"key,", "type,", "ratio,", "array," に応じた値でなくてはなりませんが、レジスタセットの全て（doValue_ch1〜ch4）を指定する必要はなく、上記のようにその一部 (doValue_ch1 と doValue_ch3) を指定することができます。

書込み制御に対する応答メッセージは、読み出し制御とは異なり指定されたレジスタセットの値のみを返します。ただし、次のよに書き換えの指定がレジスタセットの一部 (doValue_ch1 と doValue_ch3) であっても、レジスタセットの全て（doValue_ch1〜ch4）のデータを返します。

```
{
  "timestamp": "2019-11-07T20:30:05.758+09:00",
  "request_from": "0x13(0)",
  "reply_to": "38bfe79195fc09b37b6a85b6fdc97d73", "result": true,
  "doValue_ch1": 1, "doValue_ch2": 0, "doValue_ch3": 1, "doValue_ch4": 0,
  "pulseCountReset_ch1": 1, "pulseCountReset_ch2": 0,
  "pulseCountReset_ch3": 0, "pulseCountReset_ch4": 0
}
```

ここで ”reply_to ”は、制御メッセージのハッシュ値（MD5）です。

Modbus 2 サーバー

PD Handler Modbus2 Server の下流方向制御について説明します。PD Handler Modbus2 は、PD Handler Modbus とは異なり Modbus モデリングファイルに基づきサーバーのレジスタマップが定義されるため、クラウド側からは、レジスタマップの一括読み込みと入力レジスタ (inputBits もしくは inputRegister) に対する書き込みしかできません。

レジスタマップを読み取るためには、次のように空の制御メッセージを送ります。

```
{ }
```

対する応答メッセージは、次のようになります。

```
{
  "timestamp": "2019-11-07T15:30:05.758+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、制御メッセージのハッシュ値（MD5）です。

サーバーの入力レジスタ (inputBits または inputRegisters) に値を書き込む場合は、クラウドより次のような制御メッセージを送ります。

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

ここで、write キーに与えられるオブジェクトは、Modbus モデリングファイルの outputBits または inputRegisters に定義される、"key", "type", ratio", "array" に応じた値でなくてはなりませんが、レジスタセットの全て（di00〜di03）を指定する必要はなく、上記のようにその一部 (di00) を指定することができます。

書き込み制御に対する応答メッセージは、読み出し制御とは異なり指定されたレジスタセットの値のみを返します。ただし、次のよに書き換えの指定がレジスタセットの一部 (di00) であっても、レジスタセットの全て（di00〜di03）のデータを返します。

```
{
  "timestamp": "2019-11-07T16:35:13.653+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", "di01": "0x00", "di02": "0x00", "di03": "0x00",
  "regi": [123, 456, 678, 910]
}
```

ここで ”reply_to ”は、制御メッセージのハッシュ値（MD5）です。

SW4x の下流方向制御

PD Handler SW4x の下流方向制御について説明します。

例えば、RID 0x40 に接続されている SW4230 CO2 センサーノードの高度を 1234m に設定するの場合、クラウドからは次の制御メッセージを送ります。

```
{
  "rid": "0x40",
  "cmd": "0x200102000000000000001234"
}
```

ここで、cmd キーに指定される文字列は、無線センサーネットワークの入力メッセージの CMD：に与えられるコマンド文字列です。

対する応答メッセージは、制御メッセージが無線センサーネットワークに出力され、送り先ノードより ACK が返されたことのみを示すものです。

```
{
  "timestamp": "2019-11-10T16:35:13.653+09:00",
  "reply_to": "9f266f13c400de77ec977ec172c4ee3e", "result": true,
  "request_from": "0x13(0)"
}
```

PD Agent

PD Agent は、PD Repeater から制御メッセージ（JSON 文字列）を受け取り、予め設定されたキーと値の一致を持ってユーザーが用意する実行オブジェクトもしくはシェルスクリプトを実行します。(PD Agent での比較条件は文字列型での完全一致となります。)

PD Agent はユーザー定義のコンフィグ設定を用いることで下流方向の制御機能を持たない PD Handler BLE 等と連携することを前提に用意されているアプリケーションですが、本章では PD Handler とは連携させず、PD Agent をユーザー定義デバイスとして登録し単独で動作させる方法を例にその使用方法を説明します。

ユーザー定義デバイスの登録と送受信先の設定

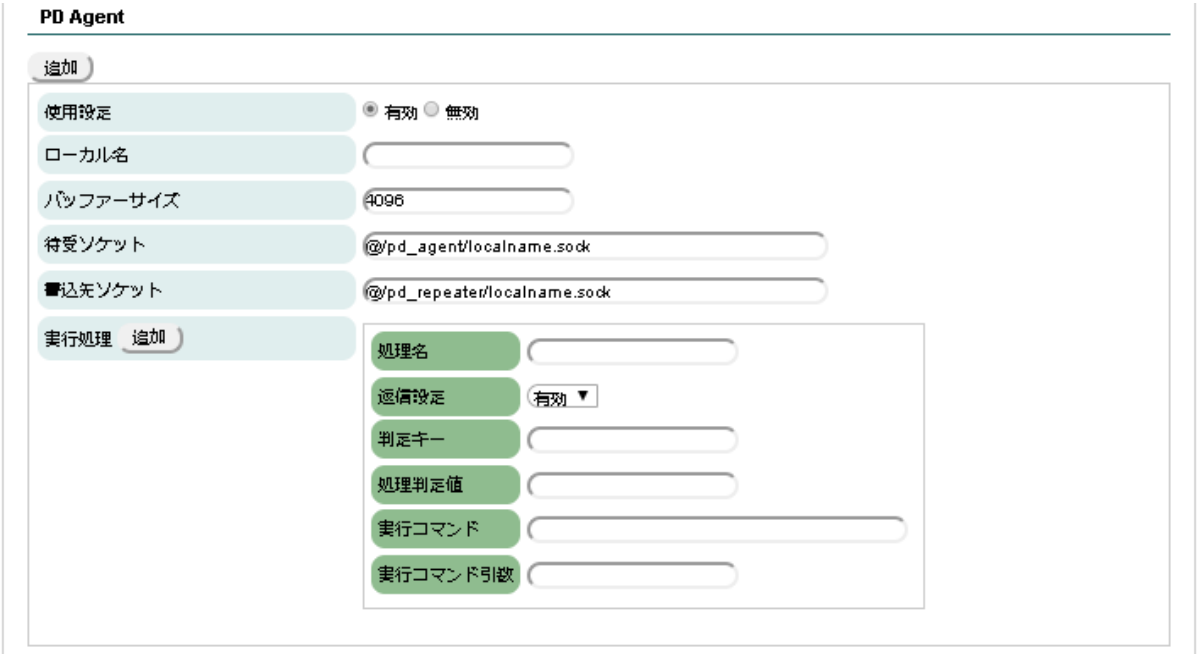
1. WEB UI の「サービス」→「基本」→「ユーザーデバイス登録」タブより、PD Agent をユーザー定義デバイスとして登録します。送受信先が複数ある場合等、複数のデバイスとして運用する場合は、複数登録します。サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。ユーザー定義のデバイスの登録とは、デバイス番号を割り振りメモ情報を付与する作業です。
2. WEB UI の「IoT データ」→「アプリ設定」タブのアプリケーション制御メニューより、PD Agent が起動されるよう設定します。アプリケーションの起動設定については、*IoT* **データ制御のアプリ設定** を参照して下さい。
3. WEB UI の「IoT データ」→「送受信設定」タブの「送受信設定」メニューより、送受信先の選択と送受信設定の内個々のデバイスに依存しない設定を行います。送受信設定については **送受信先毎の設定** を参照して下さい。
4. WEB UI の「IoT データ」→「送受信設定」タブの「デバイス設定（ユーザー定義）」メニューよりユーザー定義デバイスの送受信設定を行います。ユーザー定義デバイスの送受信設定については **デバイス設定** *(***ユーザー定義***)* を参照して下さい。

PD Agent の設定

WEB UI の「IoT データ」→「PD Agent」タブより、PD Agent の設定を行います。

初期状態では上図のようになっています。

「使用設定」を「有効」にすると設定項目が表示されます。
複数のデバイスに対する設定行う場合は「追加」をクリックし設定フォームを追加します。



「使用設定」を「有効」にすると上図のように設定項目が表示されます。

表 80: PD Agent の設定項目

設定項目	説明
使用設定	この設定を有効にするか無効にするか選択します。
ローカル名	通常はデバイスに付与されたデバイス番号とします。
バッファーサイズ	データの最大サイズを設定します。デバイス側の設定と一致させて下さい。単位はバイトです。
待ち受けソケット	制御メッセージを受け取る UNIX ドメインソケットのパス名です。 本例では、@/pd_handler/userdev_0000001.sock に設定します。 詳細は欄外に記載します。
書込先ソケット	応答メッセージを送る UNIX ドメインソケットのパス名です。 本例では、@/pd_repeater/userdev_0000001.sock に設定します。 詳細は欄外に記載します。
実行処理（追加）	以下に記載する判定キー・処理判断値・実行コマンド・実行コマンド引数を複数設定したい場合に設定フォームを追加します。
処理名	処理の名称を記載します。
送信設定	「有効」とすることで応答メッセージを送ります。実行されるコマンドに応答メッセージを返す機能が無い場合等に用います。
判定キー	「実行コマンド」に指定されるオブェクトの実行条件として評価される制御メッセージに含まれる JSON 文字列のキーを設定します。
処理判定値	「実行コマンド」に指定されるオブェクトの実行条件として評価される制御メッセージに含まれる JSON 文字列の値を設定します。
実行コマンド	「判定キー」に指定されるキーと「処理判断値」に指定される値が制御メッセージに含まれる場合に実行される実行オブジェクトもしくはシェルスクリプトのパス名を設定します。
実行コマンド引数	実行コマンドの引数を設定します。

待ち受けソケット：

制御メッセージを受け取る UNIX ドメインソケットのパス名です。

デバイスとしてユーザー定義デバイスを用い、PD Repeater から直接制御メッセージを受け取る本例では、@/pd_handler/userdev_0000001.sock に設定します。

先頭の @ は、パス名が Abstract UNIX ドメインソケット名であることを意図し、Abstract UNIX ドメインソケット名をサポートしない実行オブジェクトから待ち受ける場合
は、/tmp/pd_agent_0000001.sock 等のファイルシステムに実在するパス名とします。

書込ソケットのパス名：

応答メッセージを送る UNIX ドメインソケットのパス名です。

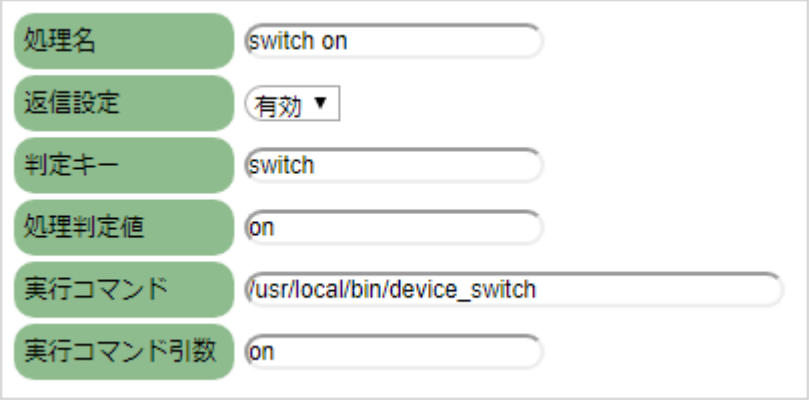
デバイスとしてユーザー定義デバイスを用い、PD Repeater へ直接応答メッセージを送る本例では、@/pd_repeater/userdev_0000001.sock に設定します。

先頭の @ は、パス名が Abstract UNIX ドメインソケット名であることを意図し、Abstract UNIX ドメインソケット名をサポートしない実行オブジェクトへ送る場合は、/tmp/ anyobject_0000001.sock 等のファイルシステムに実在するパス名とします。

PD Agent の設定と制御メッセージ

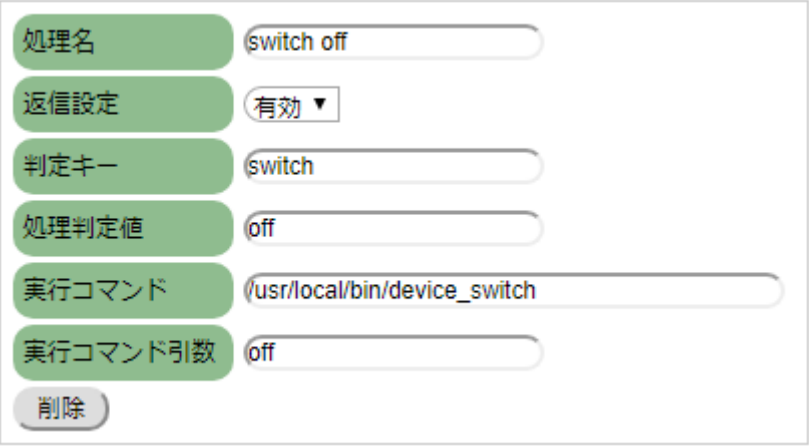
PD Agent の設定と制御メッセージの例を幾つか示します。

次の設定に対する制御メッセージは { "switch": "on" } となります。



一致する制御メッセージを受け取ると '/usr/local/bin/device_switch on' が実行されます。

次の設定に対する制御メッセージは { "switch": "off" } となります。



一致する制御メッセージを受け取ると '/usr/local/bin/device_switch off' が実行されます。

次の設定に対する制御メッセージは { "report": "do" } です。

一致する制御メッセージを受け取ると '/usr/local/bin/device_report' が実行されます。

上記の例において制御メッセージとして {"switch": "on", "report": "do" } 場合は、'/usr/local/bin/device_switch on' と '/usr/local/bin/device_report' が実行されます。

注釈:

複数の実行条件が適用されるメッセージを受信した場合、連続で処理が実行されます。
実行処理順については担保されませんので、ご注意ください。

環境変数への継承

制御メッセージ（JSON 文字列）の内、実行オブジェクトもしくはシェルスクリプトの実行条件として用いられるキーと値を含め、その値が文字列か数値であれば、それらは環境変数に設定され実行オブジェクトもしくはシェルスクリプトへ継承されます。

例えば、以上の例において { "switch" : "on", "report": "do", "page": 40 } の制御メッセージを受け取った場合、コマンドの実行においては次の環境変数が継承されます。

```
switch="on"
report="do"
page=40
```

また、制御メッセージの内容に関わらず PD Agent は、実行オブジェクトもしくはシェルスクリプトの実行において、次の値を環境変数として継承します。

表 81: 実行オブジェクトもしくはシェルスクリプトに環境変数として継承されるパラメータ

環境変数	データ型	説明
request_cloud_id	整数	*PD Repeater* **の下流方向メッセージ** に記載される Cloud ID
request_sub_id	整数	*PD Repeater* **の下流方向メッセージ** に記載される Sub ID
request_header	文字列	*PD Repeater* **の下流方向メッセージ** に記載されるヘッダー
request_md5	文字列	*PD Repeater* **の下流方向メッセージ** に記載される制御メッセージのハッシュ値
request_payload	文字列	クラウドから送られてきたペイロード
agent_localname	文字列	*PD Agent* **の設定** の「ローカル名」に設定されるローカル名
agent_bind	文字列	*PD Agent* **の設定** の「待ち受けソケット」に設定される待ち受けソケット名
agent_push_to	文字列	*PD Agent* **の設定** の「書込ソケット」に設定される書込ソケット名
agent_buffer_size	整数	*PD Agent* **の設定** の「バッファサイズ」に設定されるバッファサイズ

応答メッセージ

PD Agent **の設定** の「送信設定」が「有効」に設定されている場合は、実行ステータスとして次の応答メッセージを返します。

実行コマンドを起動した

```
{
   "time": "timestamp", "reply_to": "md5", "result": "queuing",
   "reason": "matched", "matched": {"key": "value"}
}
```

実行コマンドの実行に失敗した

```
{
   "time": "timestamp", "reply_to": "md5", "result": "faild",
   "reason": "matched", "matched": {"key": "value"}
}
```

実行コマンドの実行を終了した

```
{
   "time": "timestamp", "reply_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", "reply_to": "md5", "result": "not queuing",
   "reason": "key and value not matched"
}
```

制御メッセージが JSON 文字列でない

```
{
   "time": "timestamp", "reply_to": "md5", "result": "not queuing",
   "reason": "not JSON form"
}
```

ここで、md5 は、*PD Repeater* **の下流方向メッセージ** に記載される制御メッセージのハッシュ値、{"key": "value"} は、一致した「判断キー」と「処理判断値」です。

実行ステータスは、実行オブジェクトもしくはシェルスクリプトを呼び出す execv() の戻り値であり、実行オブジェクトもしくはシェルスクリプトの処理結果を示すものではありません。　処理の完全性を求めるのであれば応答メッセージは実行オブジェクトもしくはシェルスクリプトから agent_push_to 環境変数で継承される UNIX ドメインソケットに返すようにして下さい。

デバイス制御方式に依存した送信メッセージ構文

IoT Hub の Device Twins や AWS IoT の Device Shadow の制御においては、MQTT プロトコルのトピックを用います。

このため PD Repeater では、IoT Hub と AWS IoT について送信メッセージ (PD Reperter への入力) の構文にルールを設け MQTT プロトコルのトピックを制御します。

尚、各送信メッセージを PD Repeater が待ち受けしている Unix Domain ソケットに対して書き込むことで対応の処理が行われます。

MS Azure IoT Hub の送信メッセージ構文

IoT Hub の Device Twins と Direct Method をサポートするため、次の構文が用意されています。

Device Twins のプロパティを取得する

```
@GET <Request ID>
```

Device Twins のプロパティの更新を通知する

```
@PATCH <Request ID>;{<Reported>}
```

Direct Method に対する応答

```
@RES <Request ID>,<Status Code>;{<Any Payload>}
```

ここで、

<Request ID> : Device Twins もしくは、Direct Method のリクエスト ID

<Reported> : Device Twins の reported プロパティ（JSON オブジェクト）

<Status Code> : Direct Method の実行ステータスコード、正常時は 200.

<Any Payload> : Direct Method に対する応答ペイロード（JSON オブジェクト）

です。

これらの機能を利用するためには、IoT Hub からの下流方向メッセージ (Device Twins, Direct Method) を処理するアプリケーションの開発が必要となります。

AWS IoT の送信メッセージ構文

AWS IoT の Device Shadow をサポートするため、次の構文が用意されています。

Device Shadow の取得

```
@GET {}
```

Device Shadow の更新

```
@UPDATE {"state":{"reported": <Reported>} }
```

Device Shadow の削除

```
@DELETE {}
```

ここで、

<Reported>: Device Shadow の reported プロパティ (JSON オブジェクト)

です。

なお、モノの名前 thingName は、*AWS IoT(awsiot)* 又は *AWS IoT(awsiot)* で設定された値が用いられます。
また、AWS IoT サービスから desired プロパティを受け取るためには、AWS IoT からの下流方向メッセージを処理するアプリケーションの開発が必要となります。

MQTT サーバーの送信メッセージ構文

MQTT サーバーにおいては、メッセージ毎にトピックを柔軟に変更することができます。

MQTT トピックの変更

```
@TOPIC <topic>;{<Any Payload>}
```

MQTT トピック ユニーク ID の変更

```
@UNIQUE_ID <unique id>;{<Any Payload>}
```

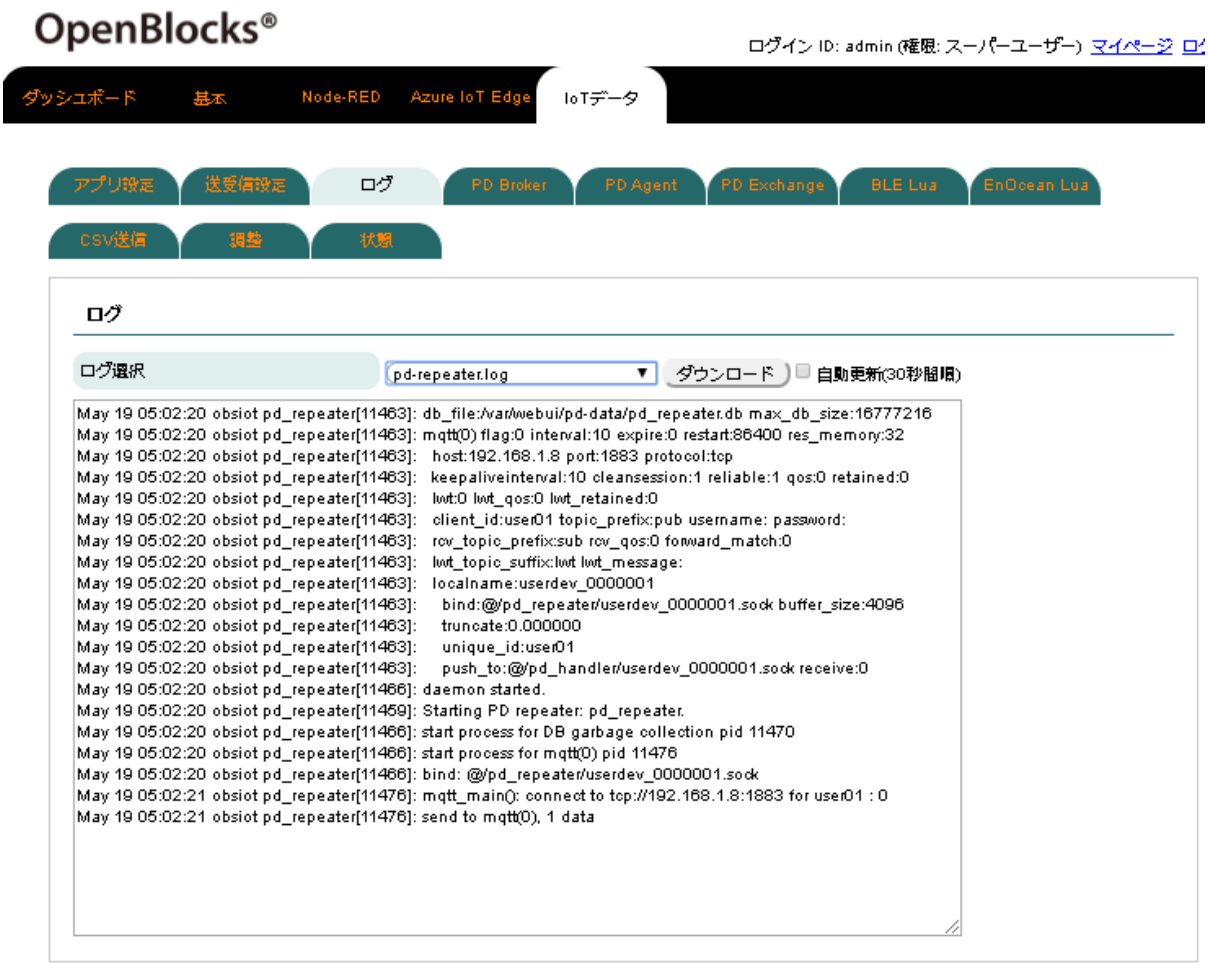
MQTT トピック プレフィックスの変更

```
@TOPIC_PREFIX <topic prefix>;{<Any Payload>}
```

20.1.14 IoT データ制御ツールログについて

IoT データ制御ツールのログについては、「IoT データ」→「ログ」タブから確認が行えます。

閲覧したいログファイルをログ欄から選択します。



ログファイルの選択を行うと、対象ログファイルの末尾が表示されます。
ダウンロードボタンを押すことによりログファイルのダウンロードが行えます。
また、「自動更新 (30 秒間隔)」をチェックすると 30 秒毎に更新されます。

20.1.15 センサーデータの確認

「IoT データ制御」→「データ表示」タブは BLE センサーデータがどのように取れているかチェックするための表示ページです。
センサーデータはセンサー毎に直近 20 件を表示します。

■ グラフ表示例

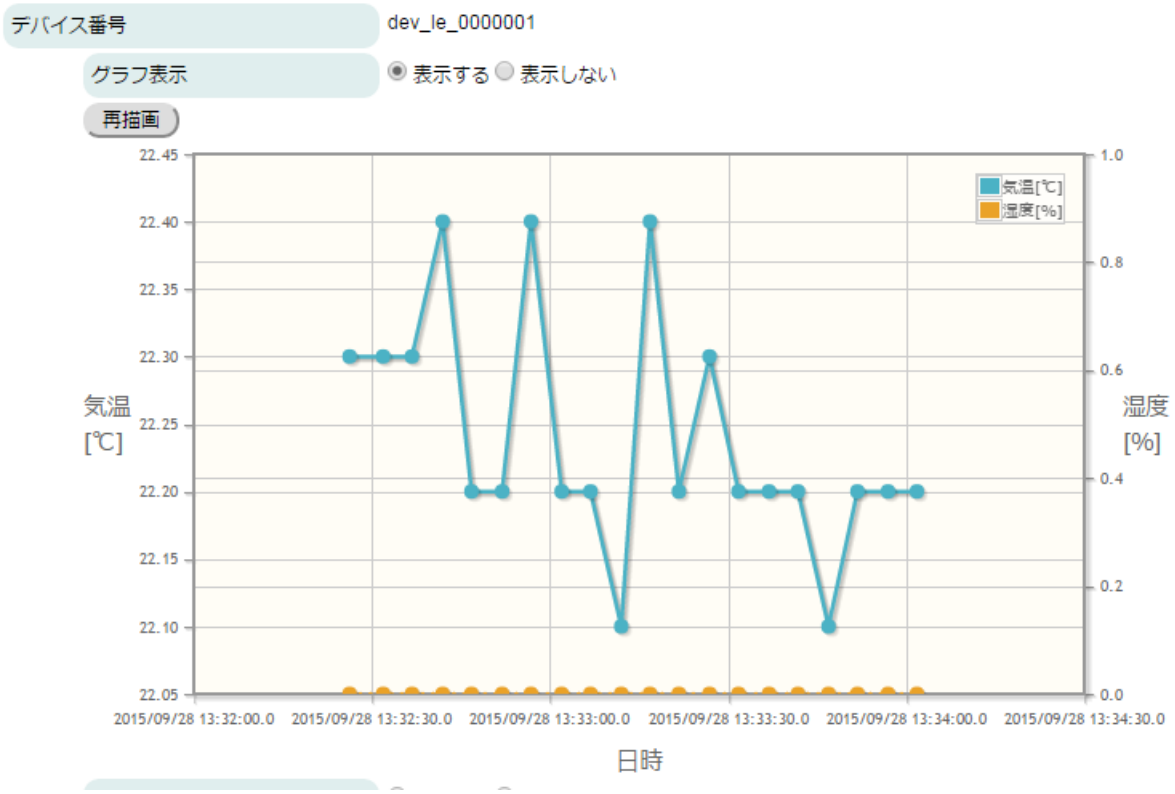


図 13: グラフ表示例

対象センサーの温度及び湿度データのグラフ描画を行います。
「再描画」ボタンで最新データから２０件のグラフ化します。
温度、湿度毎に対応していない項目の場合、”0 ℃ ”または ”0% ”として表示されます。

本グラフはセンサーのデータ収集の取得時間間隔を元に表示しています。スケールが合わせないセンサーについては、取得間隔を調整してください。

■ テーブルデータ表示例

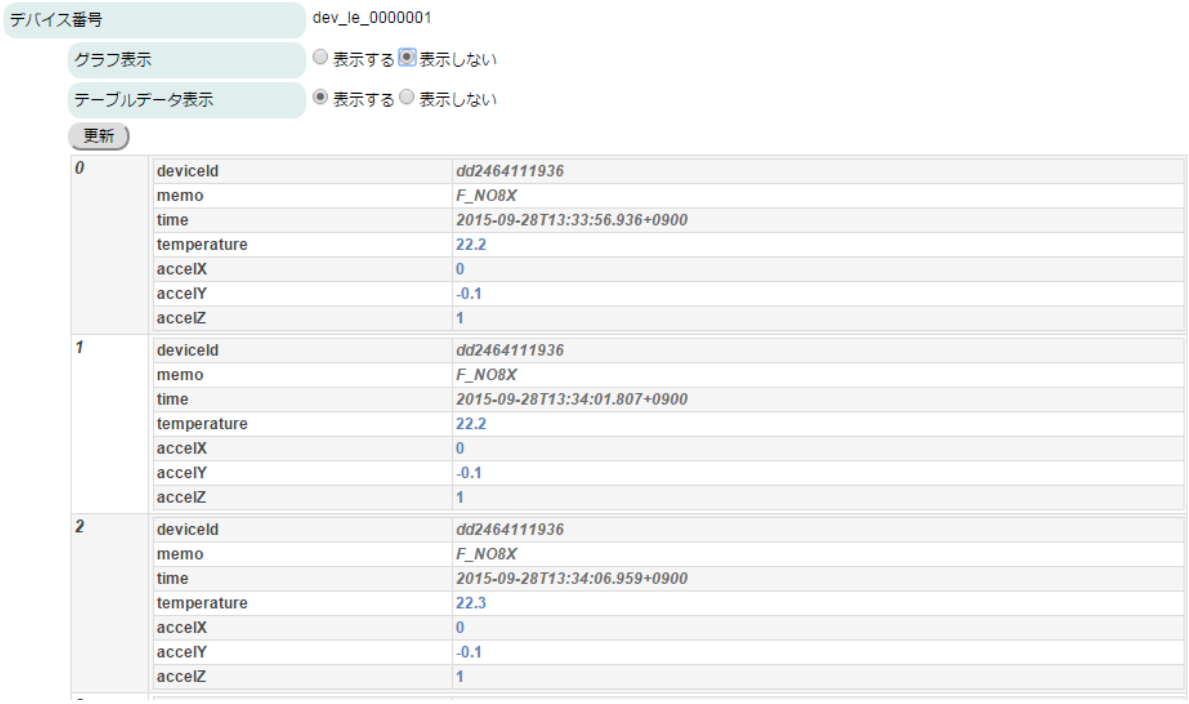


図 14: テーブルデータ表示例

「更新」ボタンで表示している内容を最新データへ更新できます。

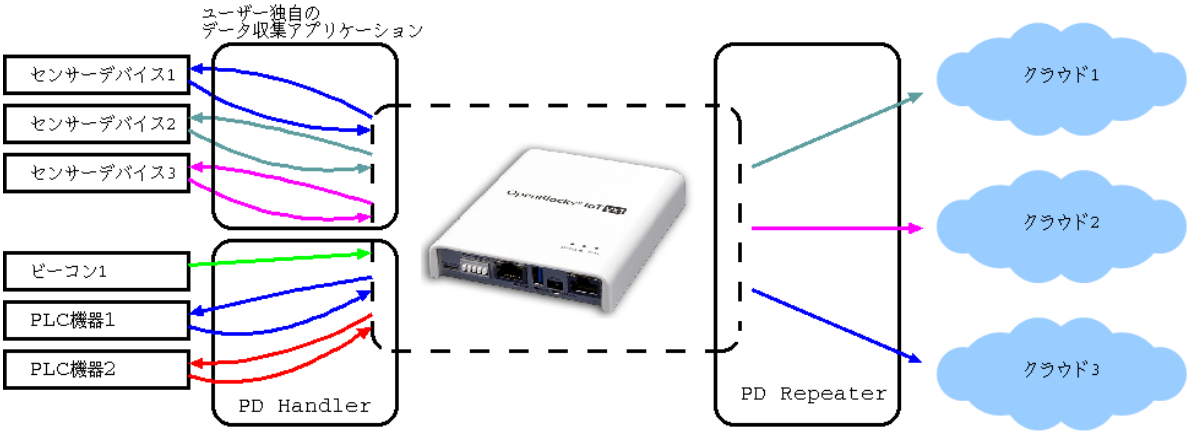
第21章　カスタマイズ

OpenBlocks シリーズの IoT データ制御機能とユーザーが独自に用意するアプリケーションとの連携方法について説明します。

21.1 独自開発データ収集アプリケーション

データの収集において弊社用意するアプリケーション (PD Handler) を用いず、各デバイス等からデータを取得するアプリケーションをユーザーが独自に開発し、PD Repeater と連携しクラウドにデータを送信する方法を説明します。

構成イメージは次の通りです。



21.1.1 ユーザー定義デバイスの登録と送受信先の設定

1. WEB UI の「サービス」→「基本」→「ユーザーデバイス登録」タブより、独自に開発したアプリケーションをユーザー定義デバイスとして登録します。

   サービス機能の基本設定については、*WEB UI* **セットアップガイド** の **サービス機能** を参照して下さい。

   ユーザー定義のデバイスの登録とは、デバイス番号を割り振りメモ情報を付与する作業です。

2. 独自に開発したアプリケーションが利用するデバイスファイルやハードウェアが弊社用意するアプリケーション (PD Handler) のそれと競合する場合は、WEB UI の「IoT データ」→「アプリ設定」タブのアプリケーション制御メニューより、該当するアプリケーションがが起動しないように設定します。

3. 独自開発のアプリケーションの起動／停止制御については **独自開発アプリケーションの起動／停止制御** を参照して下さい。

4. WEB UI の「IoT データ」→「送受信設定」タブの「送受信設定」メニューより、送受信先の選択と送受信設定の内個々のデバイスに依存しない設定を行います。

   送受信設定については **送受信設定（*PD Repeater*）** を参照して下さい。

5. WEB UI の「IoT データ」→「送受信設定」タブの「デバイス設定（ユーザー定義）」メニューよりユーザー定義デバイスの送受信設定を行います。

   ユーザー定義デバイスの送受信設定については "ref:*pd_handler_user_dev* を参照して下さい。

21.1.2 PD Repeater へのデータ書き込み

PD Repeater は WEB UI にて設定したデバイス番号を元に、抽象名前空間 (abstract) の Unix ドメインソケットを作成します。(作成する対象は送信対象を"送信する"とし、送信先が有効でかつ local 以外が設定されているデバイスです。)

この Unix ドメインソケットに対して書き込みを行った場合、書き込んだデータをクラウドへデータを送信します。

対象の Unix ドメインソケットのパス規則は以下の通りとなります。

(*IoT* **データ制御のアプリ設定** に示すユーザー定義のコンフィグを使用する場合、このパス名を任意のパス名に変更することが出来ます。)

\0/pd_repeater/<デバイス番号>.sock

以下に、'{ "x": 1 }' を各々で PD Repeater の Unix ドメインソケットに書き込みを行ったサンプルを示します。

■ コマンドラインでの書き込みサンプルは以下となります。

<デバイス番号>：userdev_0000001 として PD Repeater へ書き込んだ場合

```
# echo -n ' { "x": 1}' | socat stdin abstract-connect:/pd_repeater/userdev_0000001.
↪sock
```

注釈: socat コマンドはインストールされていません。そのため、'apt-get install socat' にてインストールしてください。

■ PHP でのスクリプトサンプルは以下となります。

<デバイス番号>：userdev_0000001 として書き込んだ場合

```
<?php
     $value = '{ "x": 1 }';
     $sock = socket_create(AF_UNIX , SOCK_STREAM , 0);
     socket_set_nonblock($sock);
     socket_set_option( $sock, SOL_SOCKET, SO_SNDTIMEO, array( "sec"=>1, "usec"=>0␣
↪) );
     socket_connect($sock , "\0/pd_repeater/userdev_0000001.sock" , 0 );
     socket_write($sock, $value);
     socket_close($sock);
?>
```

■ Node.js でのスクリプトサンプルは以下となります。

<デバイス番号>：userdev_0000001 として書き込んだ場合

```
var absocket = require('abstract-socket');
try {
        var absclient = absocket.connect('\0/pd_repeater/userdev_0000001.sock',␣
↪function() {
                console.log('connect ok');
        });
        absclient.write('{"x": 1}');
    absclient.end();
} catch(e) {
        console.log('fail');
}
process.exit();
```

注釈: abstract-socket はインストールされていません。そのため、" npm install abstract-socket"にてインストールしてください。

21.2 独自開発下流方向制御アプリケーション

クラウドから PD Repeater を介し制御メッセージを受け取り、何らかの処理をおこなうユーザーが独自に開発したアプリケーションを用いる場合の要点を説明します。

21.2.1 ユーザー定義デバイスの登録と送受信先の設定

ユーザー定義デバイスの登録と送受信先（クラウド）の設定については、**ユーザー定義デバイスの登録と送受信先の設定** に示すデータ収集用アプリケーションにおける設定と同じですが、WEB UI の「IoT データ」→「送受信設定」タブの「デバイス設定（ユーザー定義）」メニューの「受信設定」を「有効」とする必要があります。

21.2.2 PD Repeater からのデータ書き込み

PD Repeater は、クラウドから制御メッセージを受け取ると WEB UI にて設定したデバイス番号を元に定められる抽象名前空間 (abstract) の Unix ドメインソケットに *PD Repeater* **の下流方向メッセージ** に示すフォーマットで書き込みます。

Unix ドメインソケットのパス規則は以下となります。

(*IoT* **データ制御のアプリ設定** に示すユーザー定義のコンフィグを使用する場合、このパス名を任意のパスに変更することが出来ます。)

\0/pd_handler/<デバイス番号>.sock

21.3 独自開発アプリケーションの起動／停止制御

幾つかのファイルを用意することで、WEB UI による PD Repeater の起動／停止制御に独自開発アプリケーションを連動さることができます。

21.3.1 アプリケーションの登録

連動させるアプリケーションの名称を次のファイルに登録 (記載) します。：

/etc/default/obsiot-webui-ext-handler

このファイルには、1 行に 1 個のアプリケーション名を記載します。空行及びアプリケーション名の重複は許容されません。

記載例）

```
testhandler
myhandler
```

21.3.2 アプリケーションが用いるスクリプトの指定

連動させるアプリケーションが用いるスクリプトを次のファイルに設定します。：

/etc/default/<アプリケーションの名称>

ファイルにおいては次のように起動・停止・状態確認の各スクリプトを指定します。

```
bootcmd_<アプリケーション名>="<起動スクリプト>"
haltcmd_<アプリケーション名>="<停止スクリプト>"
statuscmd_<アプリケーション名>="<状態確認スクリプト>"
```

状態確認スクリプトの結果、"is running" または "RUNNING" または "RUNNIG" が出力される場合、または Status が "inactive", "failed", "dead" ではない場合、WEB UI のダッシュボードでは稼働中として認識します。

設定例）/etc/default/ testhandler

```
bootcmd_ testhandler="/etc/init.d/testhandler start"
haltcmd_ testhandler="/etc/init.d/testhandler stop"
statuscmd_ testhandler="/etc/init.d/testhandler status"
```

尚、起動スクリプト自体に ”& ”をつけて定義するのはフォーマットエラーとなります。

21.4 deb パッケージ

アプリケーションの登録 に示すアプリケーション登録処理や:ref:*data_handler_custom_rc_script* に示すファイルは、アプリケーションとその起動／停止スクリプトを含めて deb パッケージ（Debian GNU Linux のソフトウェアパッケージ）としてまとめておくことを推奨します。

deb パッケージの作成方法については Debian 公式ページをご確認ください。

deb パッケージの作成においては次の点に注意して下さい。

- **アプリケーションの登録** に示すアプリケーション登録ファイル (/etc/default/obsiot-webui-ext-handler) は、複数の deb パッケージの postinstall または postrm スクリプトによって編集されるものです。
- ログファイルを syslog 経由で吐きだすアプリケーションの場合には、postinstall にて rsyslog をリスタートしてください。(OpenBlocks シリーズで用いている syslog サービスは rsyslog です。)
- WEB UI で起動制御等が実施されますが、アプリケーションに用いるコンフィグファイルは生成されません。そのため、deb パッケージにひな形となるコンフィグファイルを入れておくことを推奨します。
- syslog 経由にてログを出力する設定等のコンフィグファイルが必要となります。また、出力先については通常の実ストレージ領域ではなく tmpfs 領域を推奨します。WEB UI にて "/var/webui/pd-logs" に tmpfs 領域を用意していますので、こちらに書き込んでください。尚、この領域に拡張子を ".log" として用意したファイルはログ確認タブから閲覧できます。| ログを大量に吐き続けた場合、ファイルサイズが大きくなり tmpfs 領域を圧迫します。| そのため、ログのローテーション設定を追加してください。tmpfs 溢れの観点からローテーション設定はファイルサイズでのトリガーを推奨とします。

21.5 アプリケーション設定の確認

アプリケーションの登録 及び:ref:*data_handler_custom_rc_script* 項にて作成した各ファイルの整合性を以下のコマンドにて確認することができます。

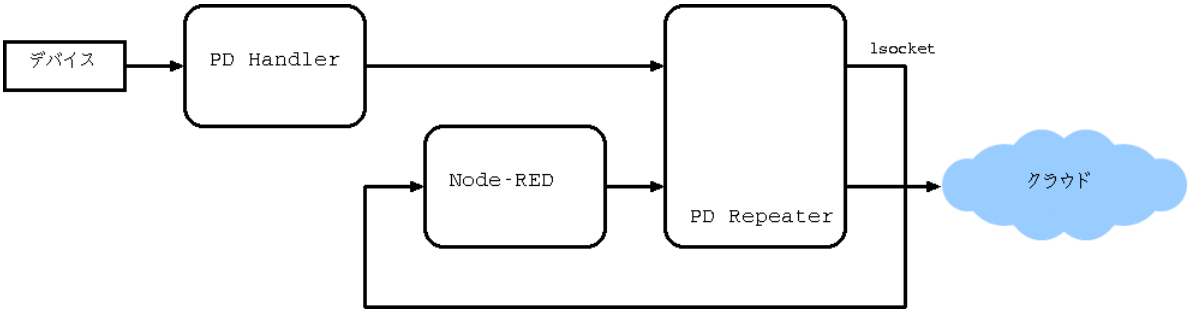
● 確認コマンド

```
# /var/webui/scripts/control-handler.sh check
```

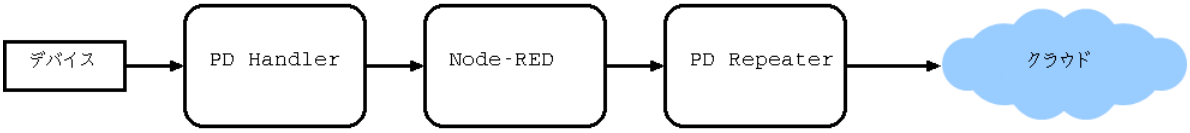
"??ERROR" から始まる行が表示される場合には設定間違い等となる為、メッセージをご確認の上、修正を行ってください。

21.6 複雑な構成の実現

PD Handler からのデータを Node-RED フィルタリングしたい場合、WEB UI の設定だけでも Node-RED をユーザー定義デバイスとすれば、PD Repeater から lsocket を用いてデータを Node-RED に渡し、Node-RED の出力を PD Handler へ返す構成が可能ですが、データの収集において弊社用意するアプリケーション (PD Handler) を用いず、各デバイス等からデータを取得するアプリケーションをユーザーが独自に開発し、PD Repeater と連携しクラウドにデータを送信する方法を説明します。



IoT **データ制御のアプリ設定** に示すユーザー定義のコンフィグを使用し UNIX ドメインソケット名を変更することで、下図のようなよりスループットを意識した構成とすることが可能です。



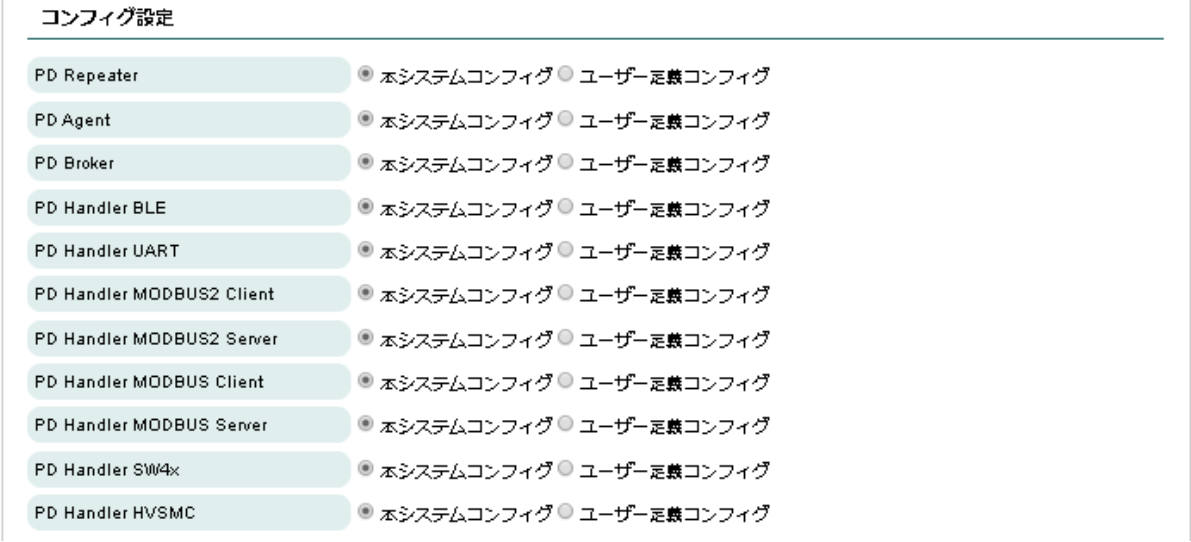
設定ファイル (コンフィグ) は JSON 形式で記載されており、UNIX ドメインソケット名は次のキーで表記されます。

表 1: アプリ起動制御の設定項目

キー	形式	
bind	文字列	データを待ち受ける UNIX ドメインソケット名。空白の場合は各アプリケーションのデフォルト値をプリフックスとし、デバイス番号を差フックスとするソケット名がもちいられる。先頭が@の場合は抽象的ドメイン名 (Abstract UNIX doamin name) であることを意図します。
push_to	文字列	データの送り先となる UNIX ドメインソケット名。空白の場合は各アプリケーションのデフォルト値をプリフックスとし、デバイス番号を差フックスとするソケット名がもちいられる。先頭が@の場合は抽象的ドメイン名 (Abstract UNIX doamin name) であることを意図します。

Beacon デバイス　（PD Handler BLE）を例に設定方法を示します。

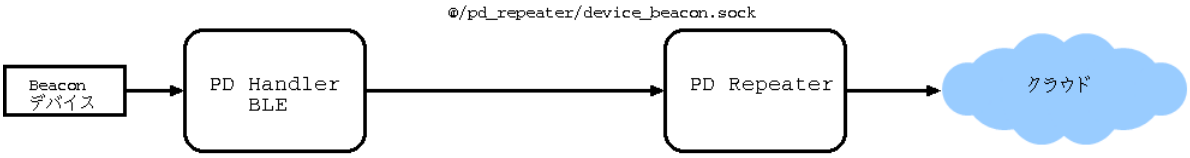
1. *IoT* **データ制御のアプリ設定** に示す「コンフィグ設定モード」が「本システムコンフィグ」であることを確認します。| WEB UI の「IoT データ」→「アプリ設定」タブより、「コンフィグ設定モード表示」をチェックして PD Handler BLE が「本システムコンフィグ」となっていることを確認します。



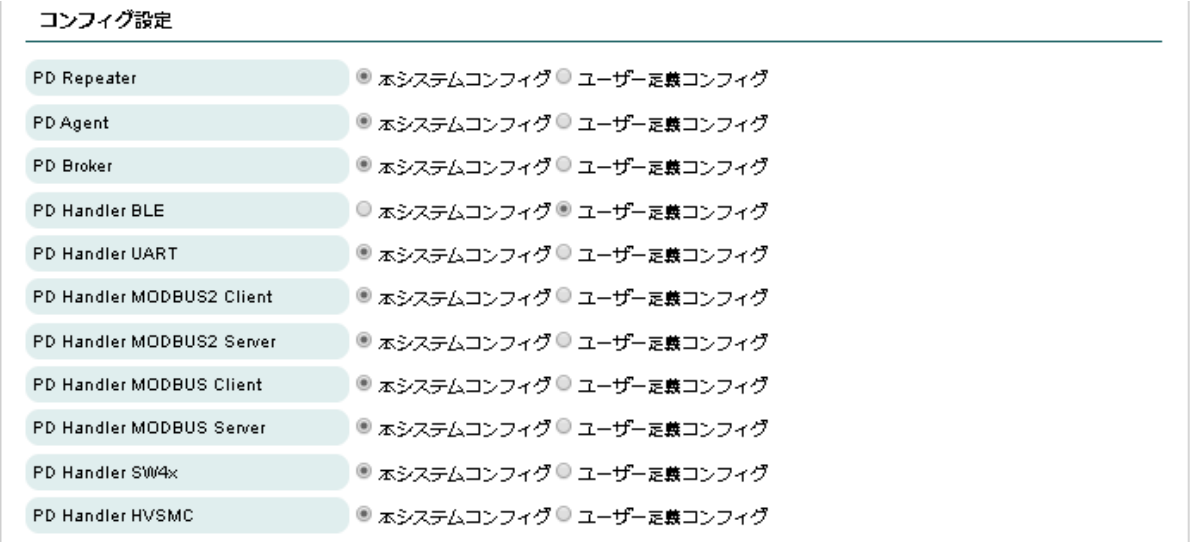
2. WEB UI の「IoT データ」→「送受信設定」タブの「送受信設定」メニューより、送受信先の設定を行います。　設定方法は:ref:*pd_repeater* を参照して下さい。

3. WEB UI の「IoT データ」→「送受信設定」タブの「ビーコン送信設定」メニューより、ビーコンを受信し送信する設定を行います。設定方法は:ref:*pd_handler_beacon* を参照して下さい。

4. ここまでの設定で下図のように PD Repeater は、UNIX ドメインソケット @/pd_repeater/device_beacon.sock でデータを持ち受け、PD Handler BLE は @/pd_repeater/device_beacon.sock にデータ書き込むよう設定されます。| ここで UNIX ドメインソケット名の '@' はソケットが抽象的ドメイン名 (Abstract UNIX doamin name) であることを意図するものです。



5. 「コンフィグ設定モード」を「ユーザー設定コンフィグ」に切り替えます。| WEB UI の「IoT データ」→「アプリ設定」タブより、「コンフィグ設定モード表示」をチェックして PD Handler BLE を「ユーザー設定コンフィグ」に切り替えます。



6. WEB UI により作成されたシステムコンフィグをユーザーコンフィグとしてコピーし、UNIX ドメインソケット名を @/pd_repeater/device_beacon.sock から @/node-red/device_beacon.sock に変更します。

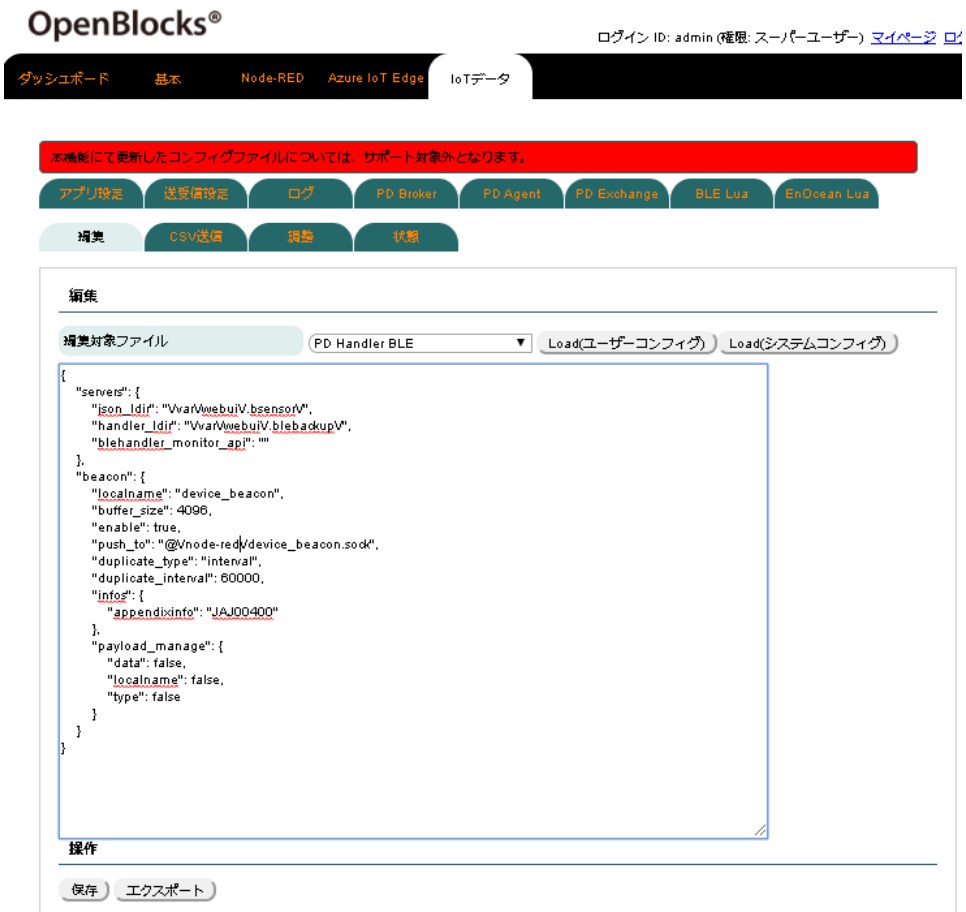
6-1. WEB UI の「IoT データ」→「編集」タブより、「編集対象ファイル」として「PD Handler BLE」を選択します。

6-2.「Load(システムコンフィグ)」をクリックします。テキストエリアに表示された設定情報（JSON 文字列）をコピーします。

6-3.「Load(ユーザーコンフィグ)」をクリックします。２項で取得した設定情報（JSON 文字列）をテキストエリアにペーストします。

6-4. "push_to" キーの値を"@/node-red/device_beacon.sock"に変更します。

6-5.「保存」をクリックします。



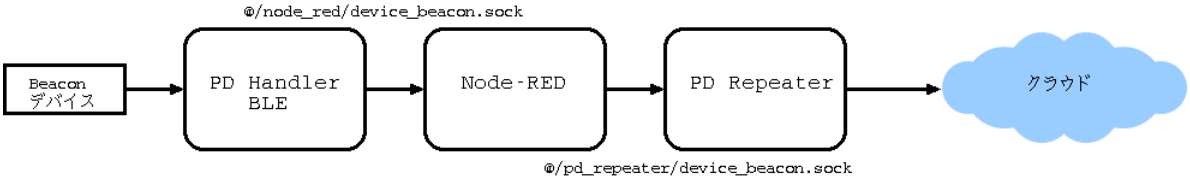
7. Node-RED の UNIX ドメインソケット入力 ipc-in ノードのプロパティを設定します。Node-RED の利用方法については *Node-RED* **スターターガイド** を参照して下さい。

   ipc-in ノードを配置し、プロパティの「Path」を"/node-red/device_beacon.sock" に設定し、「Abstract Flag」をチェックします。

8. Node-RED の UNIX ドメインソケット出力 ipc-out ノードのプロパティを設定します。

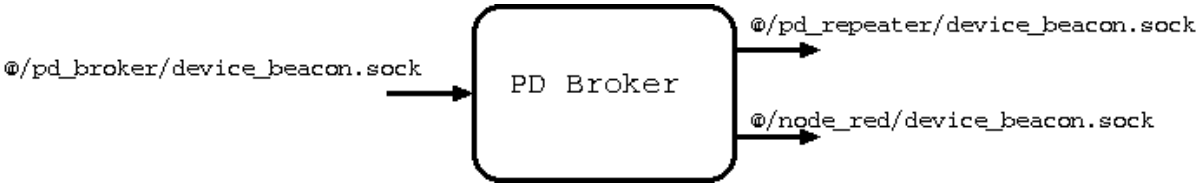
   ipc-out ノードを配置し、プロパティの「Path」を"/pd_repeater/device_beacon.sock"に設定し、「Abstract Flag」をチェックします。

以上の設定で PD Handler BLE は @/node-red/device_beacon.sock にデータ書き込み、Node-RED は @/node-red/device_beacon.sock でデータを待ち受け @/pd_repeater/device_beacon.sock に書き込むように設定されます。

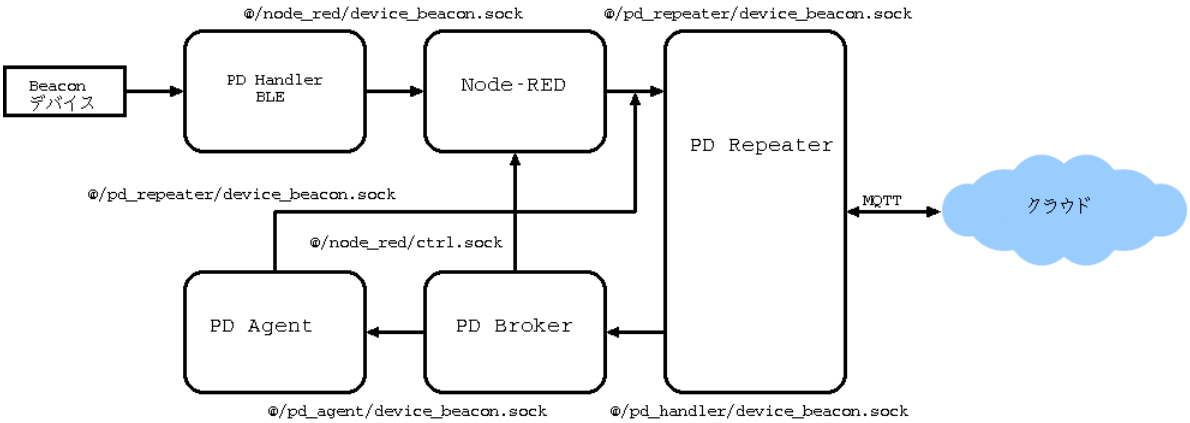


21.7 PD Broker

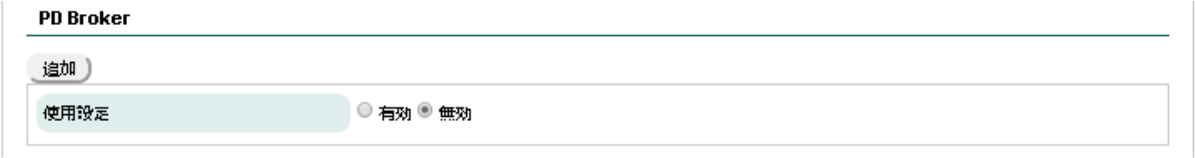
PD Broker は単一の UNIX ドメインソケット受けたデータを複数の UNIX ドメインソケットに送る、接続構成をカスタマイズするためのアプリケーションです。



例えば、下図のように PD Handler BLE でビーコンを受け、Node-RED でフィルタリングし、PD Repeater を介し MQTT 接続でクラウドへデータ送る構成で、同じ MQTT 接続を用いて制御メッセージを受け取り Node-RED のフィルタリングパラメータと PD Agent を制御しようとする場合等に用いられます。



WEB UI の「IoT データ」→「PD Broker」タブから PD Broker の設定を行うことができます。|「PD Broker」タブが表示されていない場合は、*IoT* **データ制御のアプリ設定** を参照し、PD Broker を「使用する」に設定して下さい。



初期状態では上図のようになっています。

「使用設定」を「有効」にすると設定項目が表示されます。
複数のデバイスに対する設定行う場合は「追加」をクリックし設定フォームを追加します。

「使用設定」を「有効」にすると上図のように設定項目が表示されます。

表 2: PD Broker の設定項目

設定項目	説明
追加	複数の待受けソケットを設定する場合にクリックします。２個目以降の入力フォームが表示されます。
バッファーサイズ	データの最大サイズを設定します。単位はバイトです
	待受けソケット：データを待受る UNIX ドメインソケットのパス名を設定します。先頭が '@' の場合は抽象的ドメイン名となります。
書込先設定	「追加」をクリックすると２個目以降の入力フォームが表示されます。
適応トピックフィルタ	待受けソケットから受け取ったメッセージのヘッダーに指定されたトピックと一致する場合にのみ、書込先ソケットにデータを送ります。空欄の場合、トピックによるフィルタリングは行われません。
JSON キー	待受けソケットから受け取ったメッセージに指定された JSON キー が含まれている場合にのみ、書込先ソケットにデータを送ります。空欄の場合、JSON キーによるフィルタリングは行われません。
書込先ソケット	データの送り先となる UNIX ドメインソケットのパス名を設定します。先頭が '@' の場合は抽象的ドメイン名となります。

21.8 Lua 拡張

21.8.1 BLE Lua

PD Handler BLE の C 言語版 (pd-handler-ble-c) では、対応していない非コネクション型の BLE センサーの対応が可能となっています。

WEB UI の「IoT データ」→「BLE Lua」タブから、pd-handler-ble-c にて対応したいセンサー用の Lua ファイルをアップロードすることにより対応可能です。

表 3: BLE Lua メニューの操作工億

設定項目	説明
更新	Lua ファイルのアップロード先のファイル一覧表示を更新します。
削除	ファイルを選択し、ボタンを押すことで対象ファイルを削除します。
ダウンロード	選択したファイルをダウンロードできます。
実行権付与	選択したファイルに実行権を付与します。通常では不要です。
アップロード	アップロードしたいファイルをダイアログにて選択後、押すことによりアップロードを行います。 *1

注釈:

1. アップロード後に、プロセスの再起動が必要となります。

警告: アップロード先に Lua ファイル以外が存在している場合には正常に動作しません。そのため、ファイル内容については skelton.lua を参考に作成してください。

21.8.2 EnOcean Lua

PD Handler UART の EnOcean 機能では、対応していない EnOcean デバイスへの対応が可能となっています。

WEB UI の「IoT データ」→「EnOcean Lua」タブから、対応したいセンサー用の Lua ファイルをアップロードすることにより対応可能です。

表 4: EnOcean Lua メニューの操作工億

設定項目	説明
更新	Lua ファイルのアップロード先のファイル一覧表示を更新します。
削除	ファイルを選択し、ボタンを押すことで対象ファイルを削除します。
ダウンロード	選択したファイルをダウンロードできます。
実行権付与	選択したファイルに実行権を付与します。通常では不要です。
アップロード	アップロードしたいファイルをダイアログにて選択後、押すことによりアップロードを行います。 *1

注釈:

1. アップロード後に、プロセスの再起動が必要となります。

警告: アップロード先に Lua ファイル以外が存在している場合には正常に動作しません。そのため、ファイル内容については skelton.lua を参考に作成してください。

21.9 調整

PD Repeater や各 PD Handler にて一時的な書き込みや、頻繁な書き込み用にメモリ上のファイルシステム (tmpfs) を用いています。

この部分はデフォルトでサイズを設定していますが、WEB UI の「IoT データ」→「調整」タブから調整が可能になっています。

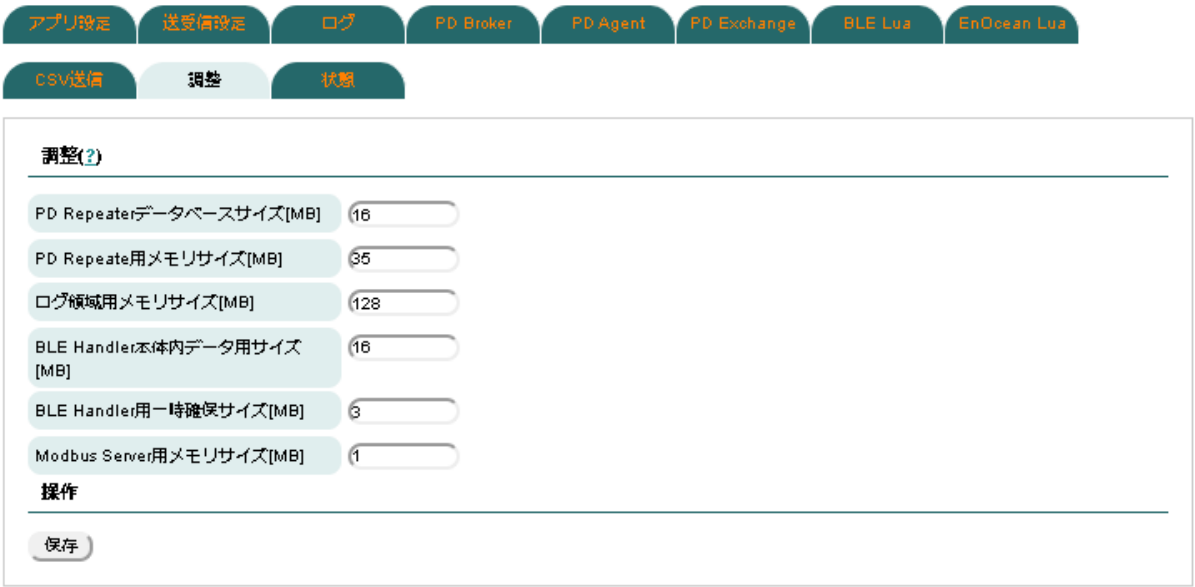


表 5: 調整メニューの設定項目

設定項目	説明
PD Repeater データベースサイズ [MB]	PD Repeter が作成するデータベースの最大サイズの指定です。16 ～ (認識メモリ / 8 - 3) [MByte] の範囲で指定可能です。
PD Repeater 用メモリサイズ [MB]	PD Repeter がデータベースの書き込みに用いる tmpfs のサイズ指定です。35 ～ (認識メモリ / 4) [MByte] の範囲で指定可能です。
ログ領域用メモリサイズ [MB]	PD Repeter 及び各 PD Handler 等のログ書き込みに用いる tmpfs のサイズ指定です。128 ～ 256 [MByte] の範囲で指定可能です。
BLE Handler 用本体内データ用サイズ [MB]	PD Handler BLE の本体内のデータ書き込み先の tmpfs のサイズ指定です。16 ～ 64 [MByte] の範囲で指定可能です。
BLE Handler 用一時確保サイズ [MB]	PD Handler BLE の一時保存領域に用いる tmpfs のサイズ指定です。3 ～ 16 [MByte] の範囲で指定可能です。
Modbus Server 用メモリサイズ [MB]	PD Handler Modbus Server の一時保存領域に用いる tmpfs のサイズ指定です。1 ～ 16 [MByte] の範囲で指定可能です。

保存ボタンを押すことに反映されます。

警告:

- PD Repeater のデータベースサイズが前回時点の物より小さいサイズを指定した場合、データベースファイルを削除します。そのため、重要なデータが存在しない事を確認の上、設定を変更してください。
- PD Repeater のデータベースとメモリサイズは以下である必要があります。:

  (データベースサイズ x 2 + 1) < (メモリサイズ)

  設定変更時に、エラーとなる場合にはご参考にしてください。

21.10 CSV データ送信機能

CSV ファイルを JSON データに変換し、PD Repeater や Node-RED の Unix Domain Socket に対して書き込みを行うことができます。

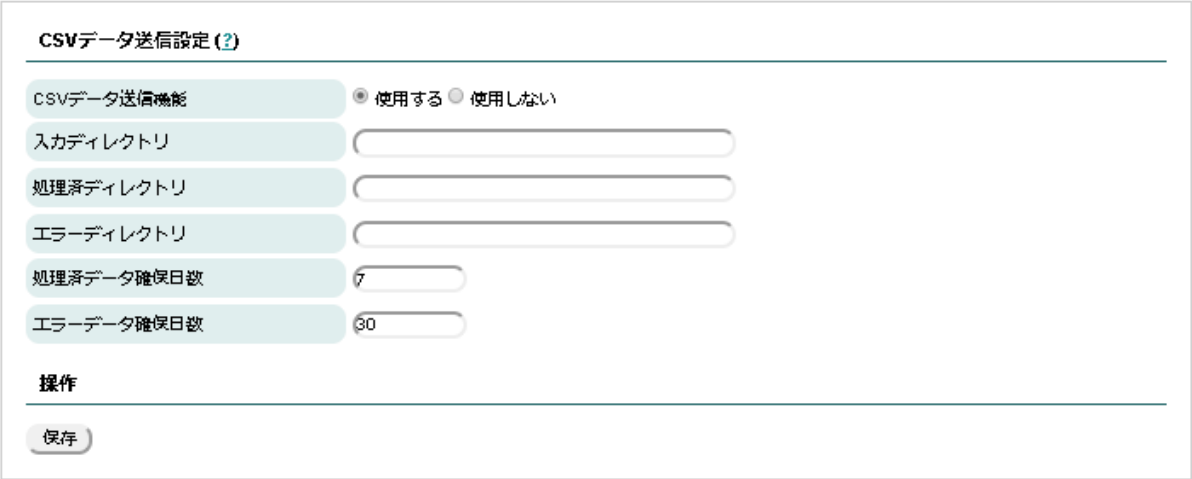
本機能では、対象のディレクトリにファイルを配置した際に自動で変換し書き込みを行います。尚、一時的に有効にしていない状態でのファイル配置時にも処理が行うよう 30 分に 1 度処理を行います。

WEB UI の「IoT データ」→「CSV 送信」タブから設定可能です。また、本機能はダッシュボードのプロセス状況の IoT 制御 (IoT データ) の「起動」、「停止」にも連動しています。

(ただし、30 分に一度の定期実行は連動していません)

■ CSV データ送信設定

初期状態では上図のようになっています。



「使用設定」を「有効」にすると上図のように設定項目が表示されます。

表 6: CSV データ送信設定の設定項目

設定項目	説明
CSV データ送信機能	本機能の使用設定となります。使用する場合には「使用する」を選択してください。
入力ディレクトリ	本機能の入力とする CSV ファイルのディレクトリの指定です。
処理済ディレクトリ	本機能にて処理を行った CSV ファイルの退避先ディレクトリの指定です。
エラーディレクトリ	本機能の処理時にエラーとなった CSV ファイルの移動先ディレクトリの指定です。
処理データ確保日数	処理済ディレクトリに配置したファイルの確保しておく日数の指定です。
エラーデータ確保日数	エラーディレクトリに配置したファイルの確保しておく日数の指定です。

保存ボタンを押すことに反映されます。

また、処理データ及びエラーデータは各確保日数経過後に削除されます。そのため、自動で削除したくない場合には、確保日数を大きくし定期的にクリーンナップ処理を行ってください。

■ CSV 変換設定

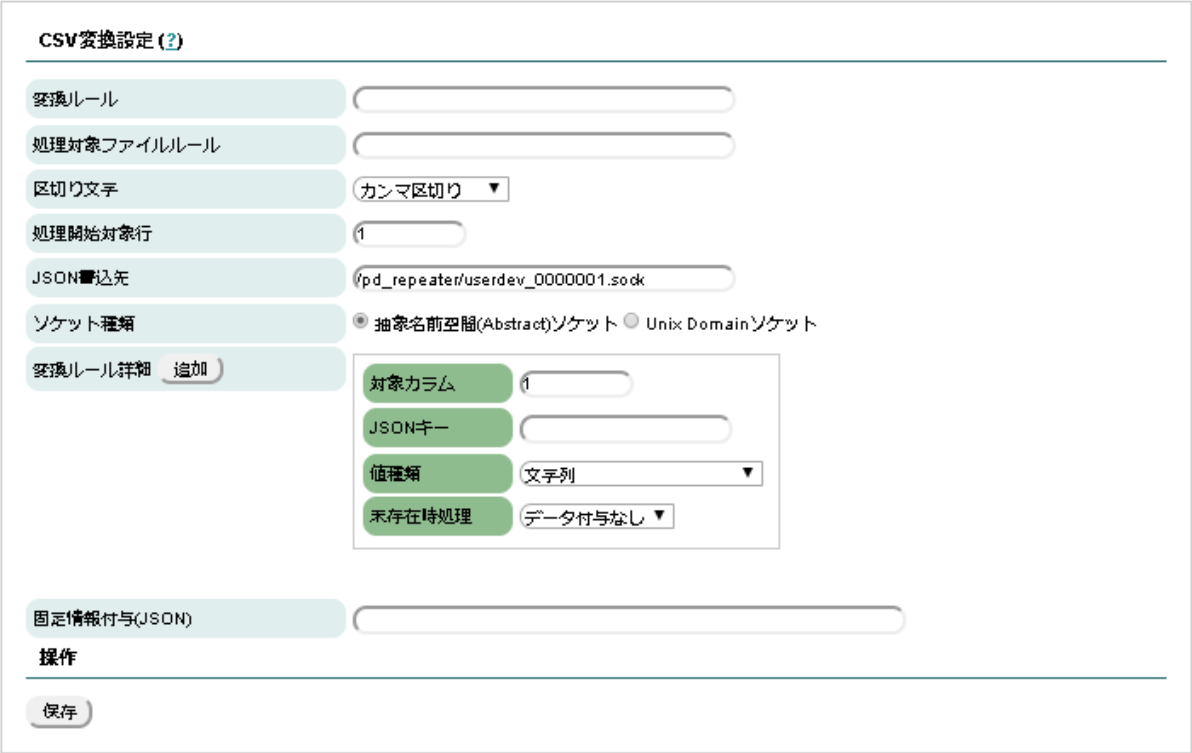

CSV変換設定 (?)
変換ルール
処理対象ファイルルール
区切り文字
カンマ区切り
処理開始対象行
1
JSON書込先
/pd_repeater/userdev_0000001.sock
ソケット種類
抽象名前空間(Abstract)ソケット
Unix Domainソケット
変換ルール詳細
追加
対象カラム
1
JSONキー
値種類
文字列
未存在時処理
データ付与なし
固定情報付与(JSON)
操作
保存

表 7: CSV 変換設定の設定項目

設定項目	説明
変換ルール	CSV の変換ルールをユニークに扱うための目ルール名を設定します。
処理対象ファイルルール	正規表現に処理対象のファイル名を指定します。
区切り文字	処理対象のファイルの区切り文字を選択します。サポートしているのはカンマ区切りまたはスペース区切りとなります。
JSON 書込先	CSV を JSON へ変換した際に書き込む先の Unix Domain Socket を指定します。
ソケット種類	書き込み先の Unix Domain Socket が Abstract ソケットか通常の Unix Domain ソケットかを選択します。
変換ルール詳細	
• 対象カラム	CSV のレコードの何カラム目を取り込むかの指定を行います。
• JSON キー	取り込み対象のカラムに対して設定する JSON キーを指定します。
• 値種類	値の取り込み時に設定するデータの型を指定します。尚、”ISO8601 日付フォーマット ”は自動変換を行います。変換に失敗した場合は、値は設定されません。
• 未存在時処理	対象カラムが存在しない場合の処理を設定します。尚、空文字設定は値種類が ”文字列 ”の場合に適用されます。
固定付与データ (JSON)	CSV を JSON へ変換した際に固定情報として付与したいデータを JSON 形式にて記載します。尚、本項目に設定可能な JSON は第一階層が Hash 情報となります。| Ex.) : {"add_info": "sample", "add_value": 1 }

警告: 処理対象ファイルルールの正規表現は PHP における preg_match 関数を適用しています。そのため、WEB 等にて preg_match の正規表現の確認を推奨いたします。尚、入力フォーム部に設定する箇所は preg_match 関数の以下の部分となります。

```
preg_match('/<入力フォーム部>/', $filename)
```

■ CSV 変換ルール一覧

CSV 変換設定の一覧が表示されます。
編集ボタンを押すことにより、CSV 変換設定のフォームに設定している内容が入力されます。
また、削除ボタンを押すことにより設定したルールを削除することができます。

第22章　補足事項

22.1 データ送信量及び回線速度について

ビーコンやデバイスからの情報取得量に対して、データ送信が遅い場合には、OpenBlocks シリーズ内のバッファーに情報が溜まっていきます。この場合、データ送信部の改善を行わない場合には溜まり続けてしまう為、バッファーデータを確認しインターバルや取得時間間隔等を調整してください。

注釈: バッファーデータは「サービス」→「状態」タブにてバッファーファイルのサイズを確認できます。

22.2 PD Repeater への書き込みデータフォーマット

PD Repeater は JSON 形式のデータを扱うことを前提としていますが、接続先によっては、JSON 形式でない文字列を扱うことができます。

表 1: 非 JSON 形式の文字列に対応可能な接続先

クラウド	非 JSON 形式対応
PD Exchange	
MQTT サーバー	○
Watson IoT for Device	
Amazon Kinesis	○
WEB サーバー	
AWS IoT	○
MS Azure Event hubs	○
Watson IoT for Gateway	
MS Azure IoT Hub	○
Toami for DOCOMO	
ドメインソケット	○
KDDI IoT クラウド Standard	
PH 社独自仕様 WEB サーバー	
Google IoT Core	○
TCP	○
AWS IoT[Websocket]	○
MS Azure IoT Hub[Websocket]	○
SoftBank スマ可視専用クラウド	○

22.3 PD Repeater へのデータの書き込みサイズ

PD Repeater へのデータの書き込みサイズはデバイス毎に変更することが可能ですが、大きくすればするほどメモリを消費しパフォーマンスも低下するため、データのサイズに応じ可能な限り小さくしておくことを推奨します。
また、クラウドによって規定されている最大値も異なるため、使用するクラウドの仕様をご確認の上、設定して下さい。

22.4 PD Repeater のバッファーサイズ

PD Repeater は送信用のバッファーとして一時溜めこみを行う為、DB にバッファーとして書き込みます。各調整を行っていない場合、DB のサイズ上限のデフォルトは 16Mbyte です。
このサイズを超えた場合、新しいデータは廃棄され、DB のサイズが 8Mbyte 以下 (データベースの最大サイズの半分) になるまでデータの受信は行われません。

22.5 Repeater のエラー時の再送信

ネットワークの通信状況によって、PD Repester からクラウドに対しての送信が失敗することがあります。
この時、連続 4 回失敗した場合や想定外のエラー状態が発生した場合には、1 分後に再送信処理を開始します。

22.6 独自開発アプリケーションの設定ファイルについて

ユーザー側にて作成した独自開発アプリケーションの設定ファイル（Config）作成機能は存在していません。
ユーザー様側にて各筐体に保存する必要がありますので、ファイルアップロード機能等をご使用ください。

22.7 Node-RED へのデータ経由方法について

PD Repeater から Node-RED へのデータは Unix ドメインソケット経由となります。
PD Repeater が各データにて書き込む Unix ドメインソケットのパスは以下のものとなります。

<ソケットパスプレフィックス>/<デバイス番号>.sock

PD Repeater においては、パス名の先頭を '@' とすることで Abstract Domain Socket として扱います。（**ドメインソケット** *(lsocket)* を参照して下さい。）

Node-RED 側では、対象デバイスの ipc-in ノードを入力として Flow に用意してください。

OpenBlocks シリーズにインストールされている ipc-in ノードは、Abstract Domain Socket が扱えるよう拡張されています。

パス名は ipc-in ノードのプロパティの「Path」に設定して下さい。パス名を Abstract Domain Socket として扱いたい場合は、同プロパティの「Abstract Flag」をチェックして下さい。PD Repeater とは異なり、パス名の先頭に'@' を記載する必要はありません。

22.8 BLE デバイスとして追加したビーコンについて

WEB UI に BLE デバイスとして登録し送信対象として設定したセンサーやビーコンは個別に扱われます。

この場合において、特にビーコンデバイスはビーコンの送信設定との依存が無くなります。そのため、ビーコン送信設定の制御タイプ、データフィルタ等は適用されません。

また、対応しているセンサー付きビーコンの PD Repeater へ渡すデータは、解析されたセンサーデータとなります。しかし、通常のビーコンの場合は時刻/デバイス ID/メモ情報を PD Repeater へ渡します。

22.9 WEB サーバーへのデータ送信について

PD Repeater(OpenBlocks IoT のファームウェア) 側からは、指定した URL の Endpoint に対して HTTP POST メソッドで送信します。

そのため、HTTP サーバ側では HTTP 200 番台のステータスコードを返す必要があります。

HTTP 200 番台のステータスコードを返却する際、HTTP ヘッダやペイロードで必要なものはありません。

尚、HTTP 200 番台以外のステータスが返された場合、PD Repeater(OpenBlocks IoT のファームウェア) 側ではエラーとして扱います。

22.10 Handler コンフィグユーザー設定

「IoT データ」→「アプリ設定」タブより、Handler コンフィグ設定を「ユーザー定義コンフィグ」を選択し保存することで使用するものがユーザー定義のものへと切り替わります。

ファイル編集自体については、「IoT データ」→「編集」タブから編集を適用してください。

尚、本機能を用いた場合、サポート対象外となります。

22.11 PH 社独自仕様 Web サーバー（PD Web）

HTTP を用いた双方向通信を実現するために弊社が独自に仕様を定義した Web サーバーです。

22.11.1 PD Web の概要

- **PD Web は HTTP において双方向通信を提供します。**
  - 下流方向へのペイロード転送は、上流方向への POST に対する応答メッセージとして提供されます。
  - 上流方向へ送るべきペイロードが存在しない場合、PD Repeater は、「受信ポーリング間隔」に指定される間隔で空接続を行います。
  - 下流方向へ送るべきペイロードが存在しない場合、Web サーバーは応答ヘッダーのみを返します。PD Repeater は応答メッセージが空でない場合、push_to キーに指定される下流モジュールの UNIX ドメインソケットにペイロードを転送します。
- **PD Web の認証方式は双方向のトークン認証です。**
  - PD Repeater と Web サーバーには、デバイス毎に指定された ID と鍵 (Key) の情報を持ちます。
  - PD Repeater は、リクエストヘッダに記載するバージョン番号（PD Web 仕様のバージョン番号）・ID・タイムスタンプ・ペイロードのハッシュ値 (MD5) から構成される署名対象文字列（リクエスト用）を鍵 (Key) で著名したハッシュ値 (Shignature) をリクエスト用のトークンとします。
  - Web サーバーは、バージョン番号・ID・タイムスタンプ・ペイロードのハッシュ値 (MD5) にリクエストヘッダのトークンを加えた署名対象文字列（応答用）を鍵 (Key) で著名したハッシュ値 (Shignature) を応答用のトークンとします。
  - 応答ヘッダーの Shignature もしくはペイロードの MD5 が期待される値と一致しない場合、以降、PD Repeater は応答ヘッダーに期待される Shignature が返されるまでペイロードを空とします。
  - Web サーバー一括の BASIC 認証を併用することも可能です。
- **エラーハンドリング**
  - PD Repeater と web サーバー間のエラーハンドリングは HTTP ステータスコードのみで行われます。ステータスが 200〜299 以外の場合は、応答ヘッダーの内容に関わらず下流モジュールへのペイロードの転送は行われません。
- **ペイロードのフォーマットとトランザクション管理**
  - 上流方向のペイロードのフォーマットは JSON 文字列であり、複数のメッセージをまとめて転送できるようメッセージの数に限らず最上位階層は配列となります。
  - PD Repeater の下流方向のメッセージフォーマットは第 2-5-2 章に示す通りですが、ペイロードのフォーマットは規定されていません。下流方向のペイロードのフォーマットは PD Repeater からペイロードを受け取る下流のモジュールの仕様に依存します。

- ペイロードのトランザクション（到達や再送処理）管理は、Web サーバーと下流のモジュール間で行われることを前提とし、PD Repeater はトランザクションの管理を行いません。
- 弊社が提供する PD Handler Modbus と PD Agent はトランザクション管理用途に下流方向のペイロードのハッシュ値 (MD5) を JSON 文字列 "reply_to" キーの値として返す仕様となっています。

22.11.2 PD Web の HTTP ヘッダー

PD Web で規定されている HTTP ヘッダーは次の通りです。

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	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

PD Repeater からの要求ヘッダーの例を示します。

```
Host: 127.0.01
Accept: */*
X-Pd-Web-Version: 1.0
X-Pd-Web-Id: pd_web_02
X-Pd-Web-Time: 2017-09-01T18:11:01.101+09:00
X-Pd-Web-Md5: 26b32b7bc6b4587c2ded48128e809b08
X-Pd-Web-Signature:␣
↪c91279bc0d9896745e4e12e73917aafd56c017966a89375273ba4b9be8b02096
Content-Length: 190
Content-Type: application/json;charset=UTF-8
```

PD Repeater への応答ヘッダーの例を示します。

```
HTTP/1.1 200 OK
Date: Fri, 01 Sep 2017 08:34:35 GMT
Server Apache/2.4.27 (Unix)
X-Powered-By: PHP/5.6.31
X-Pd-Web-Version: 1.0
X-Pd-Web-Id: pd_web_03
X-Pd-Web-Time: 2017-09-01T17:34:35.000+09:00
X-Pd-Web-Md5: d41d8cd98f00b204e9800998ecf8427e
X-Pd-Web-Signature:␣
↪b4e23876e866e9225e2f83cbd1df8b6387fe5da9e160b222953464b1ae87a9d3
Content-Length: 0
Content-Type: application/json;charset=UTF-8
```

22.11.3 PD Web のトークン

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。

22.11.4 Web サーバー (PHP スクリプト) の実装例

Web サーバー (PHP スクリプト) の実装例を示します。

```
<?php

/*
 * 本 PHP スクリプトは PD Repeater の PD Web を利用するための、
 * サーバ側 PHP スクリプトのサンプルです.
 *
 * 本スクリプトを動作させるためには、次の SQL 構文で作成された SQLite3 データベース
 * (スクリプト中の $db_file) が必要となります.
 *
 *     CREATE TABLE client (id TEXT, key TEXT, flags INTEGER, payload BLOB, md5␣
↪TEXT);
 *     CREATE INDEX index_client ON client (id);
 *
 * ここで、id は、各センサーデバイス毎に設定する PD Web の ID, key はトークンを
 * 作成するための鍵、contens は PD Repeater を介して各センサーデバイスへ送信する
 * JSON 文字列 (ペイロード)、
 * md5 は JSON 文字列の MD5 ハッシュ値、flags はペイロードの送信状態を示すコードで
 * 0:送信済、1:処理中、2:未送信 を意味します.
 *
```

(次のページに続く)

(前のページからの続き)

```
* 以下に初期値の設定例を示します.
*
*     INSERT INTO client(id, key, flags, payload, md5)
*                VALUES('id00', 'key00', 0, '{"any_key":"any_value"}',
*                       'd29e8a13452e5bc5218d9df7e6ea991f');"
*     INSERT INTO client(id, key, flags, payload, md5)
*                VALUES('id01', 'key10', 0, '{"any_key":"any_value"}',
*                       'd29e8a13452e5bc5218d9df7e6ea991f');"
*
* ここで、d29e8a13452e5bc5218d9df7e6ea991 は、'{"any_key":"any_value"}' の MD5 値です.
*  Linux OS では、次のコマンドで取得することができます.
*
*     echo -n '{"any_key":"any_value"}' | md5sum
*
* ペイロードを送信するには SQL の UPDATE を用いて flags を 2:未送信 に変更します.
*
*     UPDATE client SET flags = 2 WHERE id = 'id00';
*
* 勿論ペイロードと MD5 値を合わせてセットすることも可能です.
*
*     UPDATE client
*                SET flags = 2,
*                    payload = '{"any_key":"new_value"}',
*                    md5 = '94f030ebd7bed4a5ee08fc6fa75ae64e'
*                WHERE id = 'id00';
*
* 送信を終えるを PHP スクリプトは flag を 1 に更新し、'reply_to' キーを含む
* 応答ペイロードを待ちます.
*
* 応答ペイロードの例
*
*      {"reply_to":"94f030ebd7bed4a5ee08fc6fa75ae64e","result":"done"}
*
* 応答ペイロードは PD Repeater を介してペイロードを受け取る各センサーデバイス
* (のハンドラ) が PD Repeater 介して返すものです.
* PD Agent や PD Handler Modbus は 'reply_to' キーで payload の MD5 値をハンドリング
* しますが、独自のハンドラを用いる場合は、独自の応答ペイロードとすることも可能です.
*
* 応答ペイロード受け取ると PHP スクリプトは、データベース上の md5 と reply_to の値を比較し、
* flag を 0 に更新します.
* 受信ペイロードに応答ペイロードが含まれていない場合、PHP スクリプトは flag を 2 に戻し、
* 再送します.
*
*/
   $dump_file = '/tmp/dump.txt';
   $db_file = '/tmp/pd_web.db';

   /* HTTP ヘッダーの確認 */
   if (!(isset($_SERVER['HTTP_X_PD_WEB_VERSION']) &&
         isset($_SERVER['HTTP_X_PD_WEB_ID']) &&
         isset($_SERVER['HTTP_X_PD_WEB_TIME']) &&
         isset($_SERVER['HTTP_X_PD_WEB_MD5']) &&
         isset($_SERVER['HTTP_X_PD_WEB_SIGNATURE']))) {
       /* HTTP ヘッダーが不正な場合 Bad Request 400 を返す. */
       http_response_code (400);
       exit;
   }

   /* SQLite3 データーベースの読み込み */

   /* 本コードはサンプルであるため、SQL インジェクション対策を省略しています.
```

(次のページに続く)

(前のページからの続き)

```
        実運用に利用するためには $_SERVER['HTTP_X_PD_WEB_ID'] のバリデーション
        を十分行って下さい. */

    $db = new SQLite3($db_file);
    $query = sprintf("SELECT key, flags, payload, md5 FROM client WHERE id = '%s';
↪",
        $_SERVER['HTTP_X_PD_WEB_ID']);
    $results = $db->query($query);

    if(! $results) {
        /* HTTP_X_PD_WEB_ID が存在しない場合は Unauthorized 401 を返す. */
        http_response_code (401);
        $db->close();
        exit;
    }
    else {
        $row = $results->fetchArray();
        $key = $row['key'];
        $flags = $row['flags'];
        $payload_tx = $row['payload'];
        $md5_tx = $row['md5'];
    }

    /* HTTP リクエストヘッダ内の所定の文字列とデータベース上の key を用い
       signature を作成する */

    $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);

    /* HTTP 応答ヘッダ共通部の設定 */
    date_default_timezone_set('Asia/Tokyo');
    $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]);

    header('Pd_Web_Version: 1.0');
    header('Pd_Web_Id: ' . $_SERVER['HTTP_X_PD_WEB_ID']);
    header('Pd_Web_Time: ' . $timestamp);
    header('Content-Type: application/json;charset=UTF-8');

    /* リクエストヘッダの被署名文字列が VERSION.ID.TIME.MD5 で構成されているのに対し、
           応答ヘッダの被署名文字列は、VERSION.ID.TIME.MD5.SIGNATURE
       (SIGNATURE は、リクエストヘッダに含まれる文字列) で構成されている点に注意 */

    if ($signature != $_SERVER['HTTP_X_PD_WEB_SIGNATURE']) {
        /* HTTP_X_PD_WEB_SIGNATURE と signature が一致しない場合は、
           401 Unauthorized を返す. */
        header('Pd_Web_Md5: ' . md5(''));
        $hash_hmac_data = '1.0' . $_SERVER['HTTP_X_PD_WEB_ID'] . $timestamp .
            md5('') . $_SERVER['HTTP_X_PD_WEB_SIGNATURE'];
        $signature = hash_hmac ('sha256', $hash_hmac_data, $key, false);
        header('Pd_Web_Signature: ' . $signature);
        http_response_code (401);
        $db->close();
        exit;
    }
```

(次のページに続く)

(前のページからの続き)

```
$payload = file_get_contents("php://input");

if(md5($payload) != $_SERVER['HTTP_X_PD_WEB_MD5']) {
    /* payload の MD5 値 と HTTP_X_PD_WEB_MD5 が一致しない場合は、
       406 Not Acceptable を返す. */
    header('Pd_Web_Md5: ' . md5(''));
    $hash_hmac_data = '1.0' . $_SERVER['HTTP_X_PD_WEB_ID'] . $timestamp .
        md5('') . $_SERVER['HTTP_X_PD_WEB_SIGNATURE'];
    $signature = hash_hmac ('sha256', $hash_hmac_data, $key, false);
    header('Pd_Web_Signature: ' . $signature);
    http_response_code (406);
    $db->close();
    exit;
}

/* 受信ペイロードを HTTP リクエストヘッダと共に dump_file に格納する */
$fp = fopen($dump_file, 'a+');
fputs($fp, $_SERVER['HTTP_X_PD_WEB_VERSION']);
fputs($fp, "\n");
fputs($fp, $_SERVER['HTTP_X_PD_WEB_ID']);
fputs($fp, "\n");
fputs($fp, $_SERVER['HTTP_X_PD_WEB_TIME']);
fputs($fp, "\n");
fputs($fp, $_SERVER['HTTP_X_PD_WEB_MD5']);
fputs($fp, "\n");
fputs($fp, $_SERVER['HTTP_X_PD_WEB_SIGNATURE']);
fputs($fp, "\n");
fputs($fp, $_SERVER['CONTENT_LENGTH']);
fputs($fp, "\n");
fputs($fp, $payload);
fputs($fp, "\n");
fputs($fp, "\n");
fclose( $fp );

/* flags が 1 (処理中ペイロードあり) の場合、受信ペイロードから、
   'reply_to' キーの値を取得 */
if($flags === 1) {
    $json = mb_convert_encoding(
        $payload, 'UTF8', 'ASCII,JIS,UTF-8,EUC-JP,SJIS-WIN');
    $array = json_decode($json,true);
    if ($array !== NULL) {
        /* 受信ペイロードは、JSON オブジェクトの配列です. */
        $payload_count = count($array);
        for($i=0;$i<$payload_count;$i++) {
            if (isset($array[$i]['reply_to'])) {
                /* 'reply_to' キーの値と md5_tx を比較 */
                if($array[$i]['reply_to'] === $md5_tx) {
                        /* 一致している場合は flags を 0 にする */
                    $flags = 0;
                }
                else {
                        /* 一致していない場合は flags を 2 にする */
                    $flags = 2;
                }
            }
            break;
        }
        if($i == $payload_count) {
            /* 'reply_to' キーが存在しない場合は flags を 2 にする */
            $flags = 2;
```

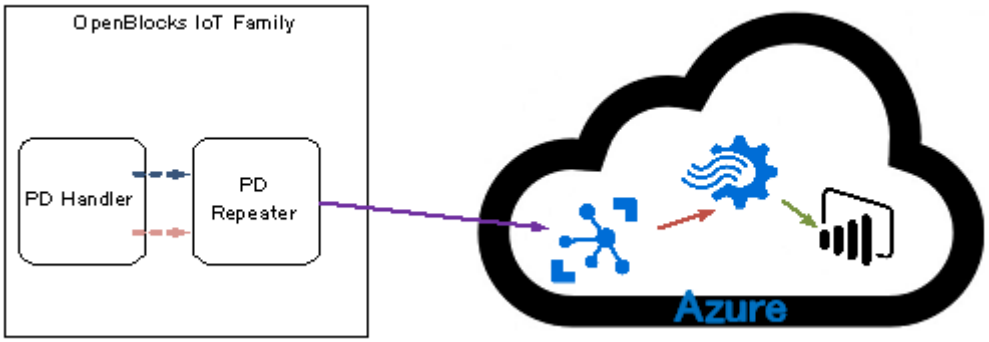
(次のページに続く)

(前のページからの続き)

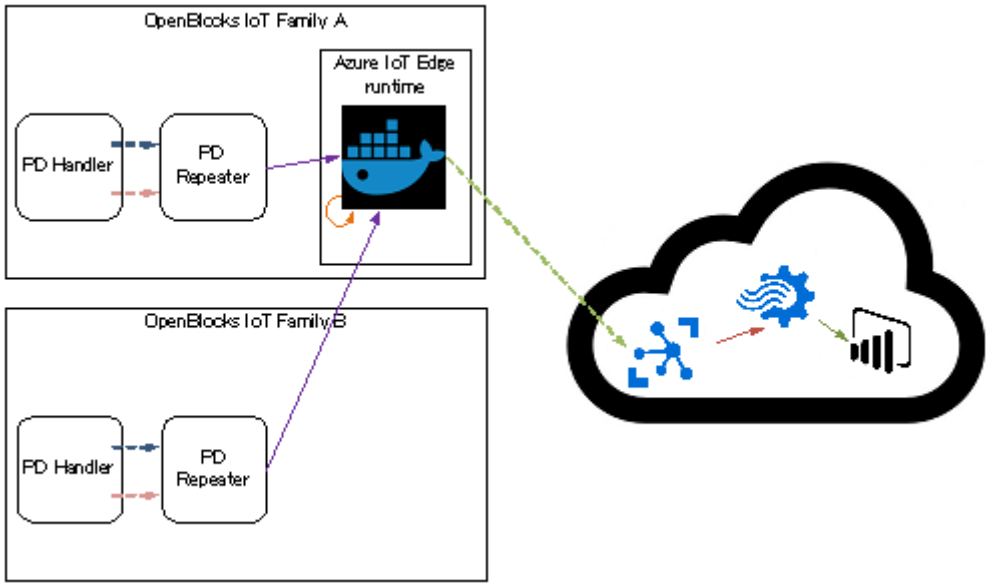
```
            }
        }
        else {
              /* JSON 文字列で無い場合、flags を 2 にする */
            $flags = 2;
        }

        /* データベースの flags を更新する. */
        $query = sprintf("UPDATE client SET flags = %d WHERE id = '%s';" ,
            $flags, $_SERVER['HTTP_X_PD_WEB_ID']);
        $results = $db->query($query);
    }

    if($flags == 2) {
        /* flags が 2 (送信ペイロードあり) の場合は、payload_tx を送り、200 OK を返す. */
        header('Pd_Web_Md5: ' . md5($payload_tx));
        $hash_hmac_data = '1.0' . $_SERVER['HTTP_X_PD_WEB_ID'] . $timestamp .
            md5($payload_tx) . $_SERVER['HTTP_X_PD_WEB_SIGNATURE'];
        $signature = hash_hmac ('sha256', $hash_hmac_data, $key, false);
        header('Pd_Web_Signature: ' . $signature);
        echo $payload_tx;
        http_response_code (200);

        /* flags を 1 (処理中ペイロードあり) に変更し、データベースの flags を更新する. */
        $flags = 1;
        $query = sprintf("UPDATE client SET flags = %d WHERE id = '%s';" ,
            $flags, $_SERVER['HTTP_X_PD_WEB_ID']);
        $results = $db->query($query);
    }
    else {
        /* 200 OK を返す. */
        header('Pd_Web_Md5: ' . md5(''));
        $hash_hmac_data = '1.0' . $_SERVER['HTTP_X_PD_WEB_ID'] . $timestamp .
            md5('') . $_SERVER['HTTP_X_PD_WEB_SIGNATURE'];
        $signature = hash_hmac ('sha256', $hash_hmac_data, $key, false);
        header('Pd_Web_Signature: ' . $signature);
        http_response_code (200);
    }

    $db->close();
    exit;
?>
```

22.12 IoT Hub / IoT Edge へのデータ送信について

PD Repeater から Azure IoT Hub に送信する場合、ホストマシン (OpenBlocks) からインターネット経由にて送信を行います。そのため、クラウドサイドにて使用するデータのみへとフィルタを行う場合、送信するデータはホストマシン (OpenBlocks 内) にて絞り込みを行うか、Azure 内の別のサービスと連携して処理を行う必要があります。



PD Repeater から Azure IoT Edge に送信する場合、ホストマシン (OpenBlocks) からホスト内に存在するコンテナまたは Azure IoT Edge が稼働している別のマシンに対してデータ送信を行います。Azure IoT Edge では、データのフィルタを行うコンテナを行うことでデータ処理が行えます。また、通常稼働ではオフライン環境での動作をサポートしており、IoT Hub へのデータ送信が必須ではありません。(コンテナのダウンロード及び構成変更等については IoT Hub との通信が必要となります。)



22.13 PD Exchange とアプリケーション、デバイスの紐づけ

PD Exchange(別売り) はセンサーデバイス毎に複数のアプリケーションへデータを供給する機能を持ちます。| 本項では PD Exchange サーバを利用する際に必要となる、PD Exchange 側へ登録したアプリケーションと、本装置に登録済みのデバイスとの紐付け方法について説明します。

「IoT データ」→「PD Exchange」タブにて以下を実施します。

表 2: PD Exchange の設定項目

設定項目	説明
デバイス番号	紐付けしたい登録されたデバイスをプルダウンメニューで選択します。
アプリケーション名	PD Exchange に予め登録されたアプリケーションを指定して「生成」ボタンを押します。PD Exchange に未登録のアプリケーション名を指定するとエラーになります。
チャネル ID 一覧	「取得」ボタンを押すと、紐付されたデバイスとアプリケーションのチャネル ID が表示されます。作成したチャネル ID を削除する場合には、「チャネル ID 削除」ボタンを押してください。

22.14 各送受信先クラウドにおけるプロトコルについて

各送受信先クラウドのプロトコルは以下の表に示す通りとなります。

表 3: 各送受信先クラウドのプロトコル

クラウド	使用プロトコル	補足
MS Azure IoT Hub	MQTT	
MS Azure IoT Hub[WebSocket]	MQTT over WebSocket	
AWS IoT Core	MQTT	
AWS IoT Core[WebSocket]	MQTT over WebSocket	
Google IoT Core	MQTT	
Watson IoT for Gateway	MQTT	
MS Azure Event hubs	AMQP	
Amazon Kinesis	REST	
Watson IoT for Device	MQTT	
Toami for DOCOMO	REST	
SoftBank スマ可視専用クラウド	MQTT	
KDDI IoT クラウド Standard	REST	
PH 社独自仕様 WEB サーバー	REST	
WEB サーバー	REST	
MQTT サーバー	MQTT	
TCP	TCP	
PD Exchange	REST	
ドメインソケット	Unix Domain Socket	

尚、REST 及び WebSocket を使うプロトコルについては HTTP プロキシに対応しています。| HTTP プロキシ環境にて使用する場合は、WEB UI の「ネットワーク」⇒「HTTP プロキシ」にて設定してください。

22.15 ダッシュボードからの起動・停止について

IoT データ制御機能をインストールしデフォルトアプリ制御部にて有効になっているアプリケーションが存在している場合、WEB UI のダッシュボードにて IoT データ制御に関するアプリケーションの動作状況が表示されます。| プロセス状況 (IoT データ) 部のボタンにより、IoT データ制御アプリケーションの一括停止や起動が行えます。

表 4: ダッシュボードの操作項目

ボタン	処理内容
起動	IoT データ制御アプリケーションを使用設定に基づき、一括起動します。
停止	IoT データ制御アプリケーションを使用設定に基づき、一括停止します。
停止 (クリア)	IoT データ制御アプリケーションを使用設定に基づき、一括停止します。また、停止後、PD Repeater のデータベースを削除します。

第VI部

Node-REDスターターガイド

第23章　はじめに

本書は、OpenBlocks に搭載可能な Node-RED の使用方法を解説しています。

搭載している Node-RED は IoT データ制御機能にて送信先の候補として用意しており、エッジコンピューティングの実装や対応していないクラウドへの対応を想定しております。

第24章　Node-RED事前準備

24.1 Node-REDのインストール

本製品出荷時では、Node-REDはインストールされておりません。そのため、WEB UIの「メンテナンス」→「機能拡張」タブからNode-REDをインストールしてください。

WEB UIの「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

インストール機能のリストから「Node-RED」を選択します。

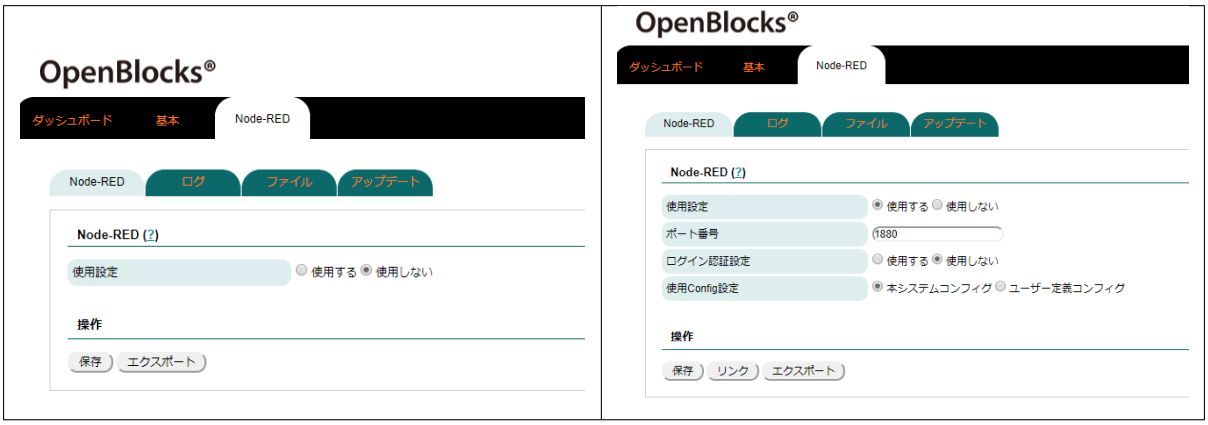
その後、インストールの「実行」ボタンを押し、インストールを行ってください。

尚、インストール完了後には反映を行うため、本体再起動が必要となります。そのため、「メンテナンス」→「停止・再起動」から本製品の再起動を行ってください。

24.2 Node-RED 起動設定

Node-RED のインストールが完了している場合、サービス WEB UI の「サービス」→「基本」タブに Node-RED の項目が表示されます。

この Node-RED のリンクを押して下さい。これにより、「Node-RED」タブへと遷移します。



「Node-RED」→「Node-RED」タブから Node-RED 自体の設定を行えます。

Node-RED	
項目	備考
使用設定	Node-RED の使用設定を行います。Node-RED を使用する場合には、「使用する」を選択してください。
ポート番号	Node-RED へアクセスするためのポート番号を設定します。通常は、デフォルトの 1880 から変更する必要はありません。
ログイン認証設定	ログイン認証を使用する場合には"使用する"を選択し、ユーザー名及びパスワードを設定し保存ボタンを押してください。
使用 Config 設定	Node-RED 自体の起動設定 Config ファイルを本システムまたはユーザー編集したものと切り替えることができます。 「ユーザー定義コンフィグ」を選択した場合、コンフィグ編集用タブが表示されます。

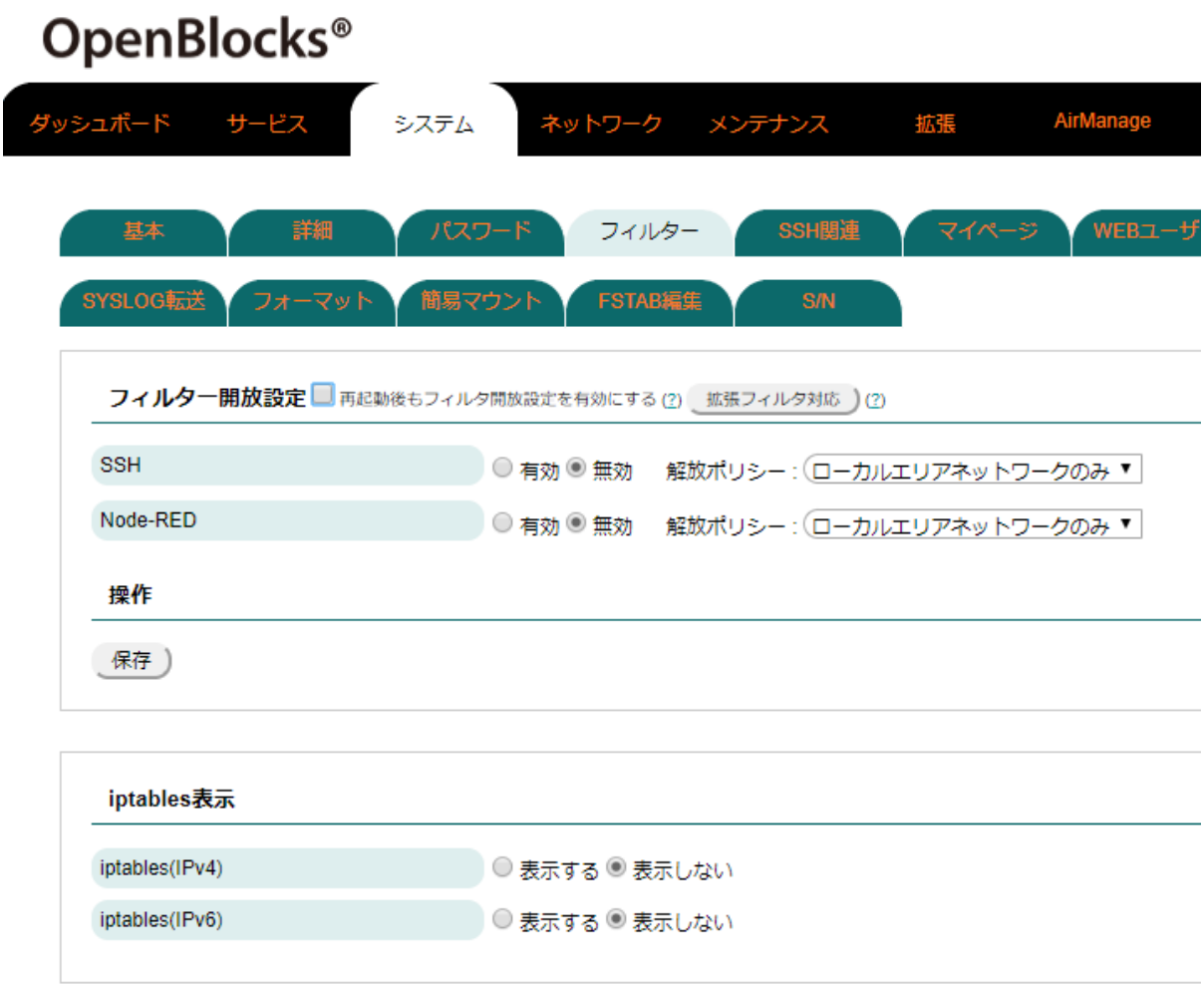
設定完了後、保存ボタンを押すことにより Node-RED が起動・停止します。

※リンクボタンを押すことによって起動中の Node-RED へアクセスすることができます。ただし、AirManage 経由の場合にはアクセスは行えません。AirManage 経由の場合には URI プロキシ機能を用いてアクセスしてください。

また、Node-RED 機能を使用する状態にて、保存ボタンを押した場合にて、フィルターが解放されていない場合には、解放確認のポップアップが表示されます。また、解放ポリシー設定ポップアップが出ますので、設定したいポリシーのボタンを押してください。

24.3 Node-RED のブラウザフィルタについて

Node-RED のインストールが完了している場合、Node-RED 自体への WEB アクセスを行う場合、Node-RED のフィルターを開放する必要があります。Node-RED 機能を使用しフィルターの開放設定を行っていない場合には、WEB UI の「システム」→「フィルター」タブにてフィルターを開放してください。



デフォルトでは Node-RED のブラウザアクセスはできないようにフィルターが適用されています。

"有効"に設定し、保存ボタンを押してください。

尚、セキュリティの観点上、Node-RED の設定完了後はフィルターを閉鎖してください。

24.4 パケットフィルタについて

Openblocks の入力方向のパケットフィルタは、WEB UI へのアクセスや時刻同期等、システムの動作に必要となるポートを除き開放されておりません。

TCP 入力ノード 等、リモートからの接続を待ち受けるノードを使用し外部から受信する場合、必要に応じて別途パケットフィルタを開放する必要があります。

パケットフィルタの操作は、/var/webui/local/bin/iptables-ext.sh ファイルを作成し WEB UI の「メンテナンス」→「フィルター」タブから、iptables 編集するスクリプトを作成してください。

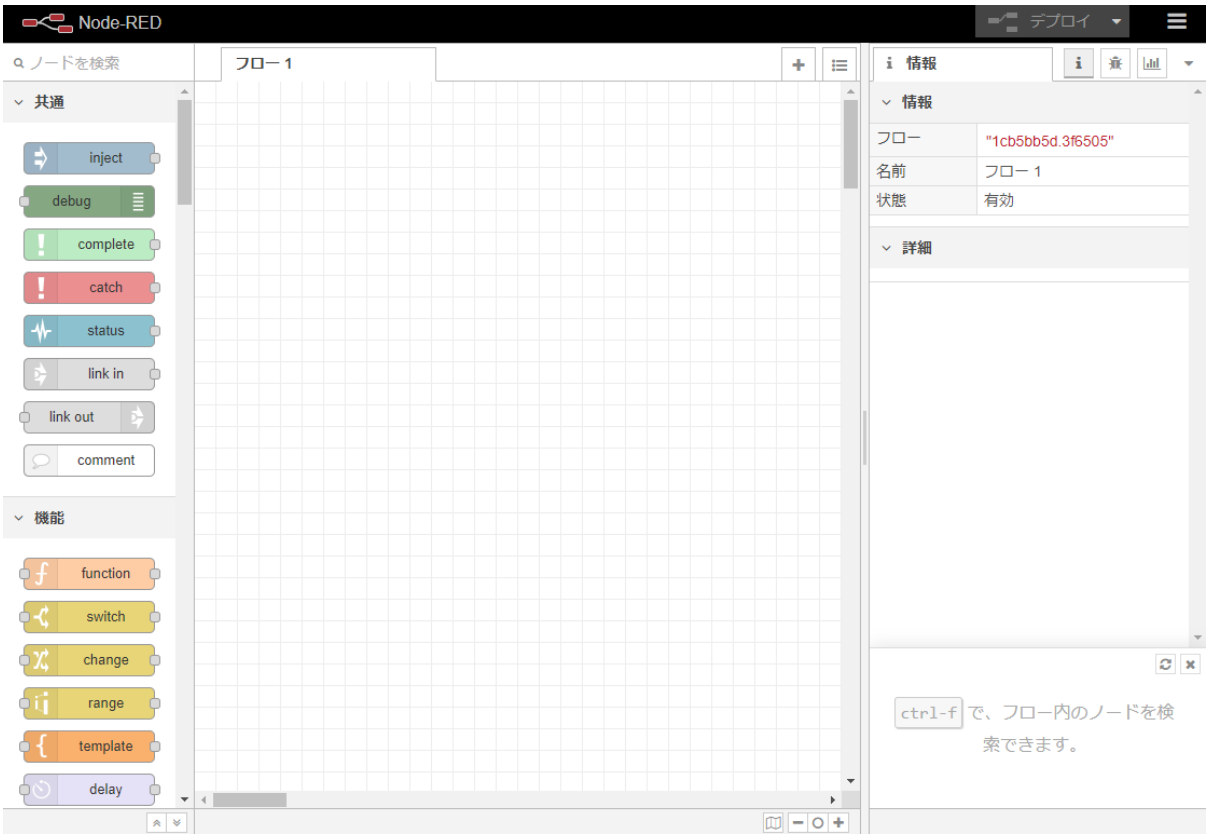
第25章　Node-REDの簡易説明

Node-RED はデフォルトで 1880 番ポートを用いてブラウザアクセス行います。

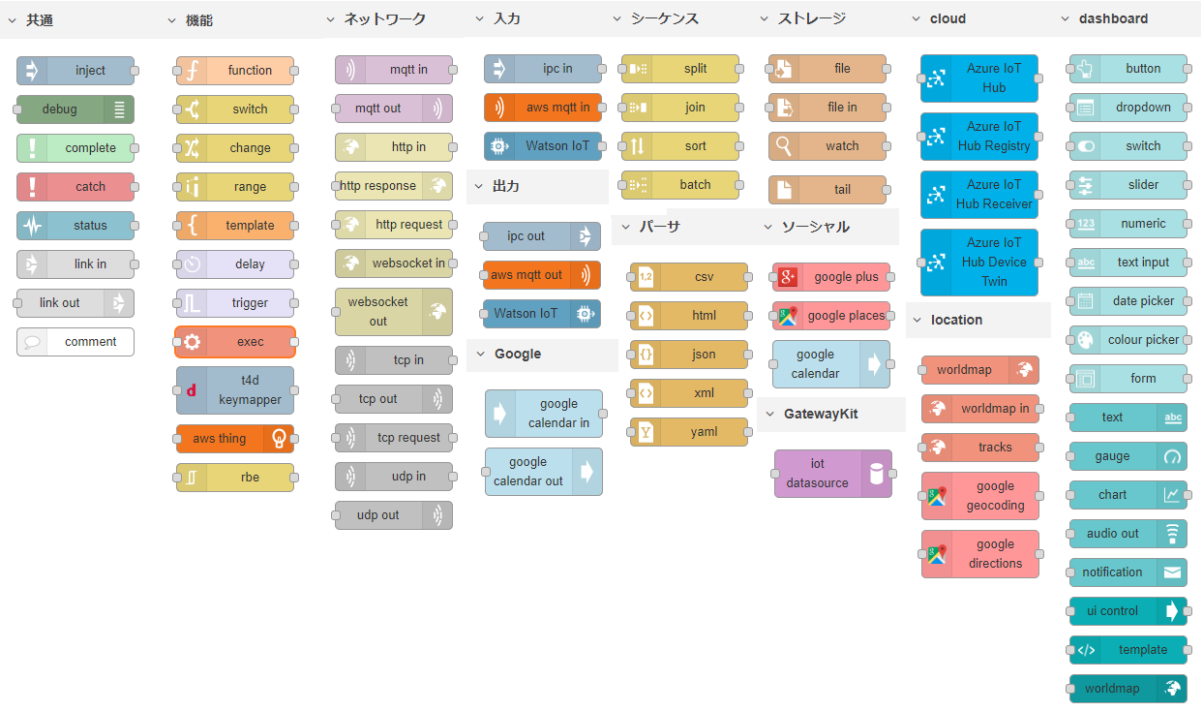
そのため、WLAN 対応の OpenBlocks シリーズでのデフォルト IP での WLAN 経由での Node-RED へのアクセスする為の URL は以下となります。OpenBlocks IX9 等での Ethernet 経由でのアクセス等については、適宜 IP アドレスを変更してください。

http://192.168.254.254:1880/

アクセスした場合、以下のような画面が初期状態では表示されます。(ログイン認証設定をしていない場合)

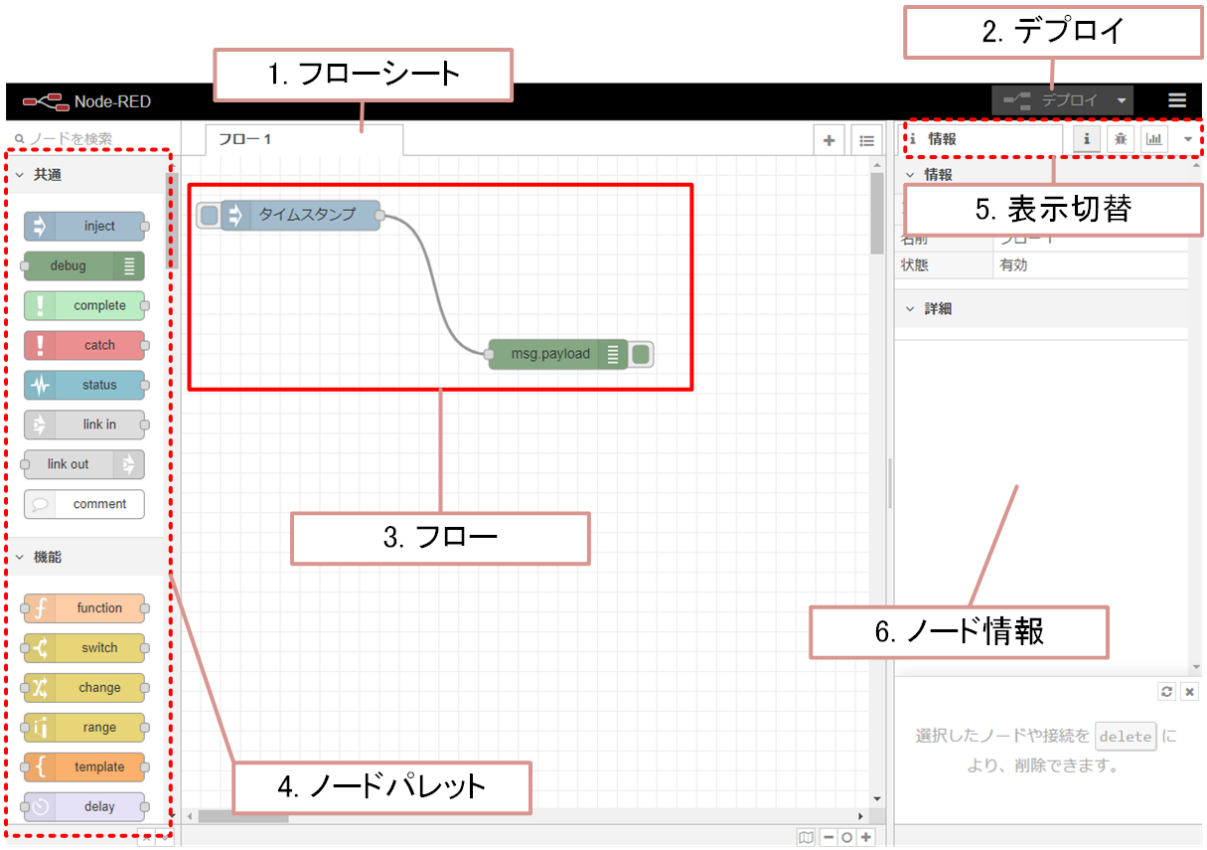


また、本製品向けにデフォルトで用意している入力・出力・処理等のノードは以下となります。



25.1 Node-RED 画面構成

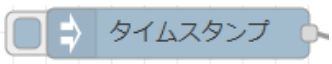
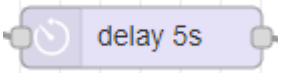
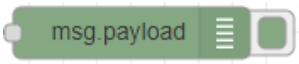
Node-RED の画面は以下のように構成されています。



#	項目	説明
1	フローシート	処理フローを記述するワークスペースです。
2	デプロイ	デプロイボタンをクリックすることでシートに記述した処理フローを有効化します。
3	フロー	ノードを配置し結線することでデータの流れ（処理フロー）を定義します。
4	ノードパレット	処理フローの構成に用いられるノードの一覧です。
5	表示切替	ノード情報・デバック情報の表示を切り替えます。
6	ノード情報	ノード情報、又はデバック情報が表示されます。

25.2 ノード種類

Node-RED では大きく分けて以下のようなコネクタ配置のノードがあります。

コネクタが右	コネクタが左右	コネクタが左
		
・イベント起動 起動条件を満たした時、データを右のノードへ送信します。	・データ処理 左からデータを受信し、処理を行います。また、処理後のデータを右のノードで送信します。	・イベント終了 左からデータを受信し、該当処理を行い終了します。

データは左から右へと処理が行われます。

25.3 共通ノード

inject	手動もしくは一定間隔でメッセージをフローに注入します。
debug	「デバッグ」タブに、選択したメッセージプロパティの値を表示します。
complete	他のノードにおけるメッセージ処理の完了を受けてフローを開始します。
catch	同じタブ内のノードが送出したエラーをキャッチします。
status	同じタブ内のノードのステータスメッセージを取得します。
link in	フロー間に仮想的なリンクを作成します。タブ上に存在する link out ノードに接続できます。
link out	フロー間に仮想的なリンクを作成します。タブ上に存在する link in ノードに接続できます。
comment	フローにコメントを記述するために利用します。

25.4 機能ノード

function	受信メッセージに対して処理を行う JavaScript コード (関数の本体) を定義します。
switch	プロパティの値によってメッセージの振り分けを行います。
change	メッセージ、フローコンテキスト、グローバルコンテキストのプロパティを変更、削除、移動します。
range	数値を異なる範囲の値に変換します。
template	テンプレートに基づいてプロパティを設定します。
delay	ノードを通過するメッセージを遅延もしくは流量を制限します。
trigger	メッセージを受信すると、別のメッセージの送信を行います。延長もしくは初期化が指定されていない場合には、2 つ目のメッセージを送信することもできます。
exec	システムのコマンドを実行し出力を返します。
t4d keymapper	入力されたメッセージを Toami for Docomo 向けのメッセージへ変換します。
aws thing	AWSIoT の ThingShadow を制御します。
rbe	例外データのノード - ペイロードの値が変化した場合だけデータを送信します。

25.5 ネットワークノード

mqtt in	MQTT ブローカに接続し、指定したトピックのメッセージをサブスクライブします。
mqtt out	MQTT ブローカに接続し、メッセージをパブリッシュします。
http in	HTTP エンドポイントを作成し、Web サービスを構成します。
http response	HTTP In ノードで受け付けたリクエストに対するレスポンスを送り返します。
http request	HTTP リクエストを送信し、レスポンスを返します。
websocket in	WebSocket により受信したデータは msg.payload に格納し出力します。
websocket out	入力された msg.payload データを WebSocket 経由で送信します。
tcp in	TCP からの入力を行います。リモート TCP ポートに接続するか、外部からのコネクションを受け付けます。
tcp out	TCP への出力を行います。リモート TCP ポートへ接続、外部からのコネクションの受け付け、もしくは、TCP In ノードで受け付けたメッセージへのリプライを行います。
tcp request	シンプルな TCP リクエストノード。msg.payload をサーバの TCP ポートに送信し、レスポンスを待ちます。
udp in	UDP 入力ノード。msg.payload にバッファ、文字列、もしくは、Base64 エンコーディング文字列を生成します。
udp out	入力された msg.payload を指定した UDP のホストとポートに送信します。

25.6 入力ノード

ipc in	Unix Domain Socket から受信したメッセージを msg.payload に送信します。
aws mqtt in	AWSIoT に接続し、指定されたトピックにサブスクライブします。トピックには MQTT ワイルドカードが含まれる場合があります。
Watson IoT	IBM Watson Internet of Things Platform からデバイスコマンドを受信します。

25.7 出力ノード

ipc out	msg.payload を Unix Domain Socket に格納し出力します。
aws mqtt out	AWSIoT に接続し、msg.topic または編集ウィンドウで指定されたトピックのいずれかに msg.payload を送信します。
Watson IoT	デバイスイベントを IBM Watson Internet of Things Platform に送信します。

25.8 シーケンスノード

split	メッセージをメッセージ列に分割します。
join	メッセージ列を結合して一つのメッセージにします。
sort	メッセージ列もしくは配列型のペイロードをソートします。
batch	指定したルールによりメッセージ列を生成します。

25.9 パーサノード

csv	CSV 形式の文字列とその JavaScript オブジェクト表現の間で双方向の変換を行います。
html	msg.payload に格納した HTML ドキュメントから CSS セレクタを使用して要素を取り出します。
json	JSON 文字列と JavaScript オブジェクトとの間で相互変換を行います。
xml	XML 文字列と JavaScript オブジェクトとの間で相互変換を行います。
yaml	YAML 文字列と JavaScript オブジェクトとの間で相互変換を行います。

25.10 ストレージノード

file	msg.payload をファイルに書き出します。書き出しは、ファイルの最後に追記もしくは既存の内容の置き換えを選択できます。
file in	ファイルの内容を文字列もしくはバイナリバッファとして読み出します。
watch	ディレクトリもしくはファイルの変化を検知します。
tail	設定したファイルの末尾を出力 (追加されたデータを監視) します。

25.11 ソーシャルノード

google plus	Google+ API を操作して、人、活動、コメントに関する情報を取得します。
google places	Google Places API を利用して、地元の施設を見つけて詳細を調べます。
google calendar	Google カレンダーの各イベントを返します。

25.12 cloud ノード

Azure IoT Hub	msg.payload を Azure IoT Hub に送信するノードです。
Azure IoT Hub Registry	デバイスを Azure IoT Hub に登録するノードです。
Azure IoT Hub Receiver	Azure Events Hub エンドポイントを介してデバイスからクラウドへのメッセージを受信するためのノードです。
Azure IoT Hub Device Twin	Azure IoT Hub Device Twin を取得するためのノードです。

25.13 GetwayKit ノード

iot datasource	入力データをダッシュボードアプリケーションへ出力します。

25.14 location ノード

worldmap	入力データを WEB 上にプロットします。（インターネット接続が必要です。）
worldmap in	Worldmap の出力 Web ページからイベントを受信します。
tracks	指定した数の前の場所に基づいてトラックを作成します。
google geocoding	Google geocording を用いて入力データをジオコーディング（住所情報と地理的座標の相互変換）し、出力します。
google directions	Google directions を用いて入力された出発地点と目的地点の経路を出力します。

25.15 dashboard ノード

button	ユーザーインターフェースにボタンを追加します。
dropdown	ユーザーインターフェースにドロップダウン選択ボックスを追加します。
switch	ユーザーインターフェースにスイッチを追加します。
slider	ユーザーインターフェースにスライダーウィジェットを追加します。
numeric	ユーザーインターフェースに数値入力ウィジェットを追加します。
text input	ユーザーインターフェースにテキスト入力ウィジェットを追加します。
date picker	ユーザーインターフェースに日付選択ウィジェットを追加します。
colour picker	ユーザーインターフェースに色選択ウィジェットを追加します。
form	ユーザーインターフェースにフォームを追加します。
text	ユーザーインターフェースに編集不可能なテキストフィールドを表示します。
gauge	ユーザーインターフェースにゲージタイプのウィジェットを追加します。
chart	ユーザーインターフェースに入力値をグラフにプロットします。
audio out	ダッシュボードの音声またはテキストを音声で再生します。
notification	msg.payload をポップアップ通知または OK / Cancel ダイアログメッセージとしてユーザーインターフェイスに表示します。
ui control	ダッシュボードを動的に制御できます。
template	テンプレートウィジェットには、任意の有効な html および角度/角度 - 材料指示を含めることができます。
worldmap	ダッシュボードのマップに「もの」をプロットします。

25.16 Google ノード

google calendar in	Google カレンダーでイベントが発生するたびにメッセージを送信します。
google calendar out	Google カレンダーにエントリを作成します。

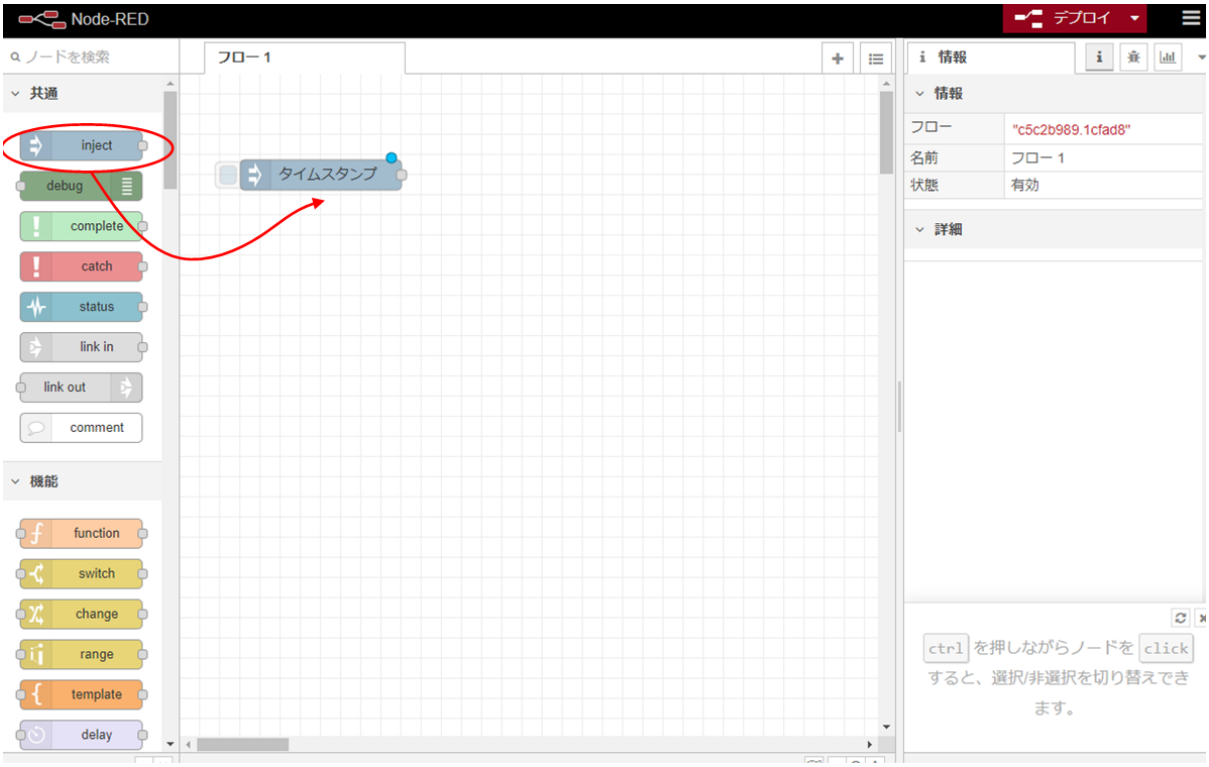
第26章　ノード操作サンプル

26.1 位置情報の可視化

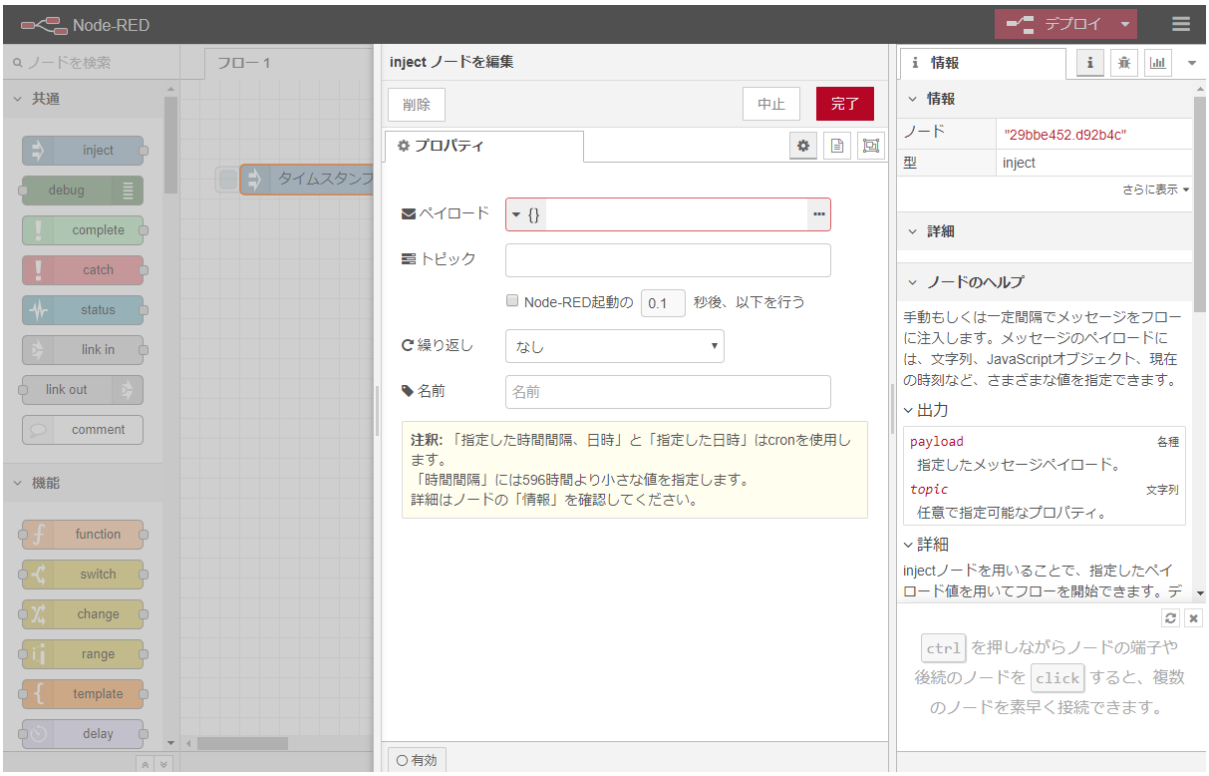
1, Node-RED にアクセスします。

http://192.168.254.254:1880/

2, 入力ノードパレットから inject ノードをドラッグしてシートにドロップします。



3, 配置した inject ノードをダブルクリックし、ノードの編集を行います。

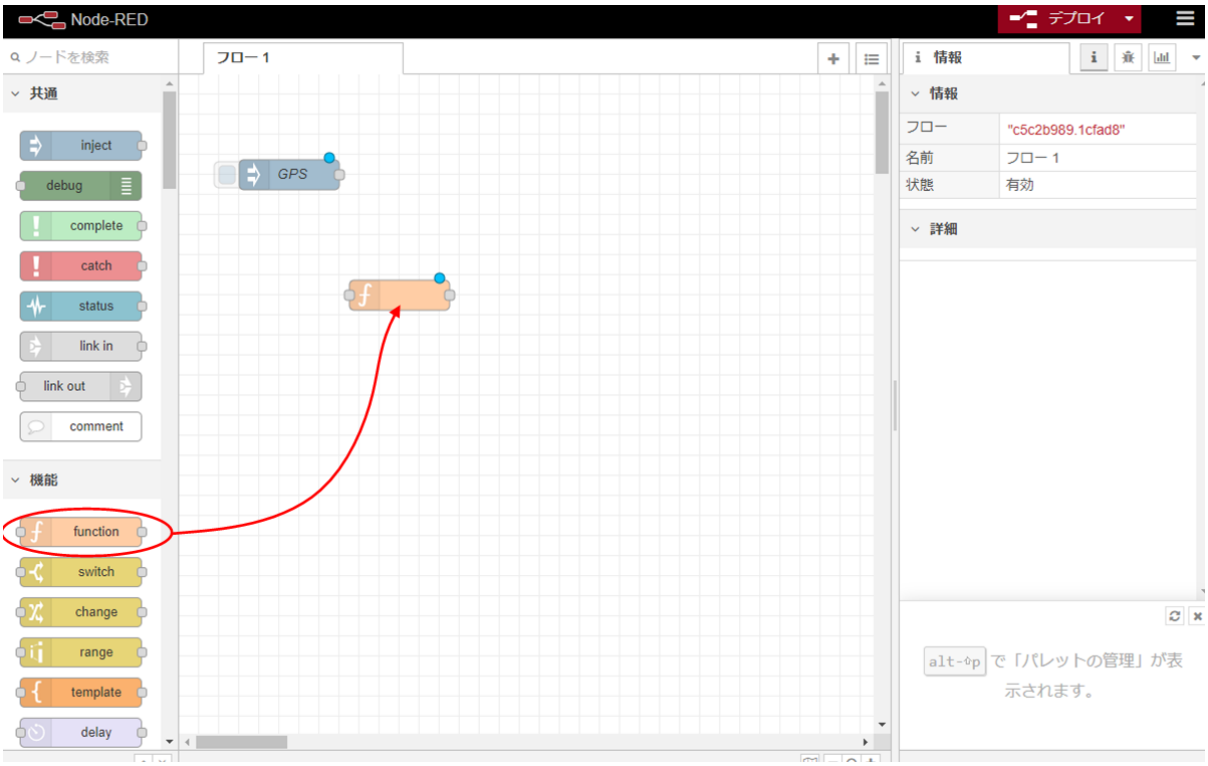


名前欄に GPS と入力します。ペイロードの種類に JSON を選択します。また、ペイロード部の『…』を選択し、表示されるウィンドウに以下を設定します。

```
{
        "latitude": 35.681167,
        "longitude": 139.767052
}
```

設定後、「完了」ボタンを押します。

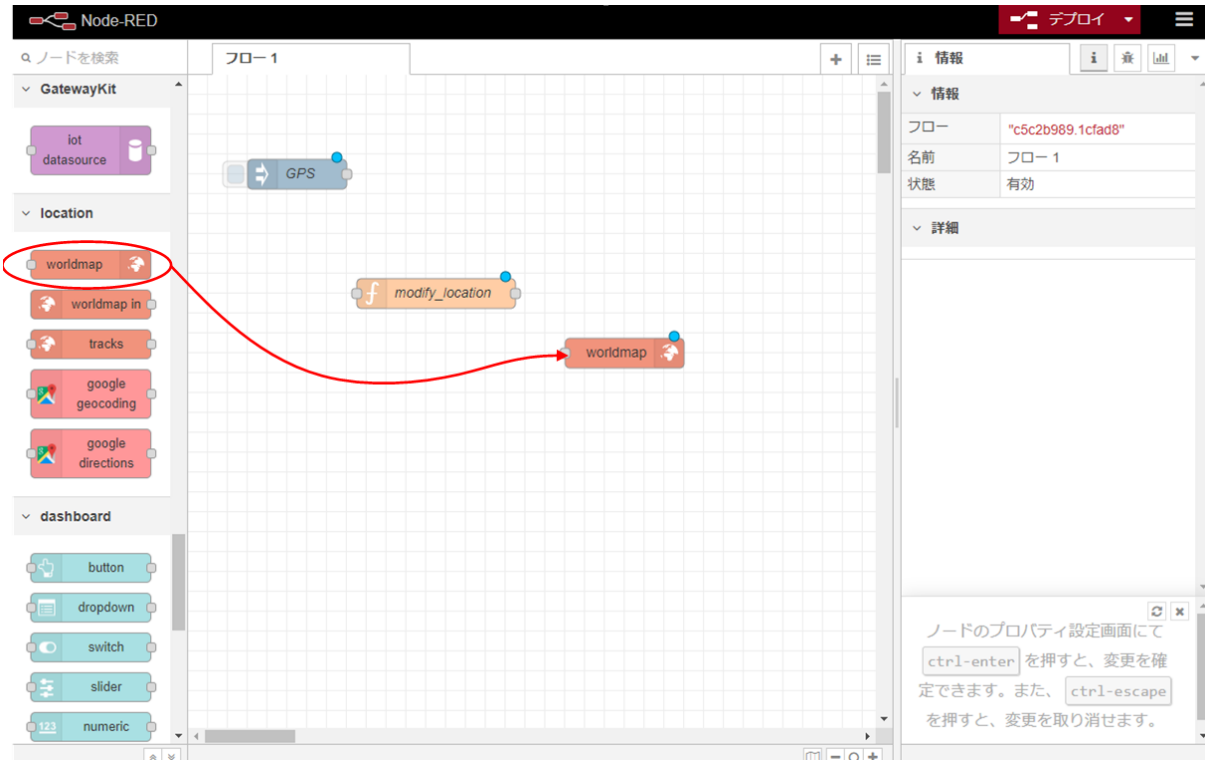
4, 機能ノードから function ノードをドラッグしてシートにドロップします。



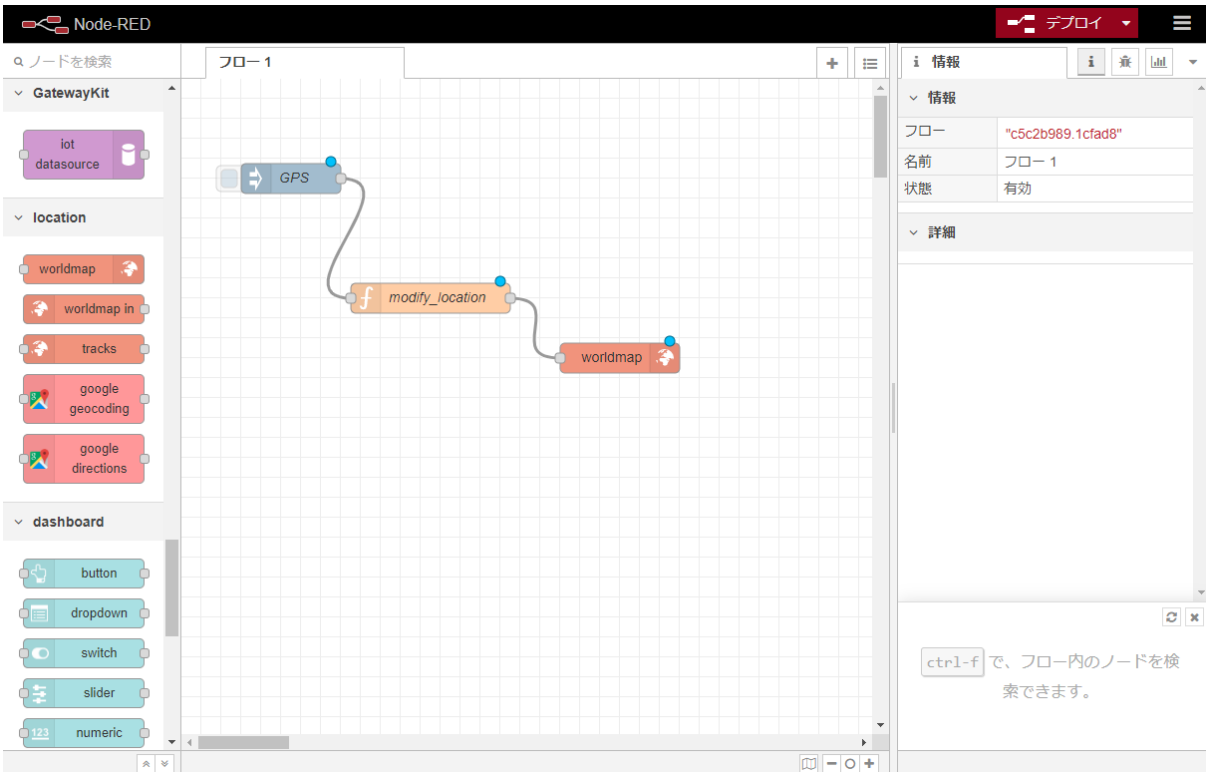
5, 配置した function ノードをダブルクリックし、名前欄に ”modify_location ”を設定します。また、コード部に以下を設定します。設定完了後、『完了』ボタンを押します。

```
//var res = JSON.parse(msg.payload);
var res = msg.payload;
msg.payload = {
        name: "Tokyo sta." ,
        lat: res.latitude,
        lon: res.longitude
} ;
return msg;
```

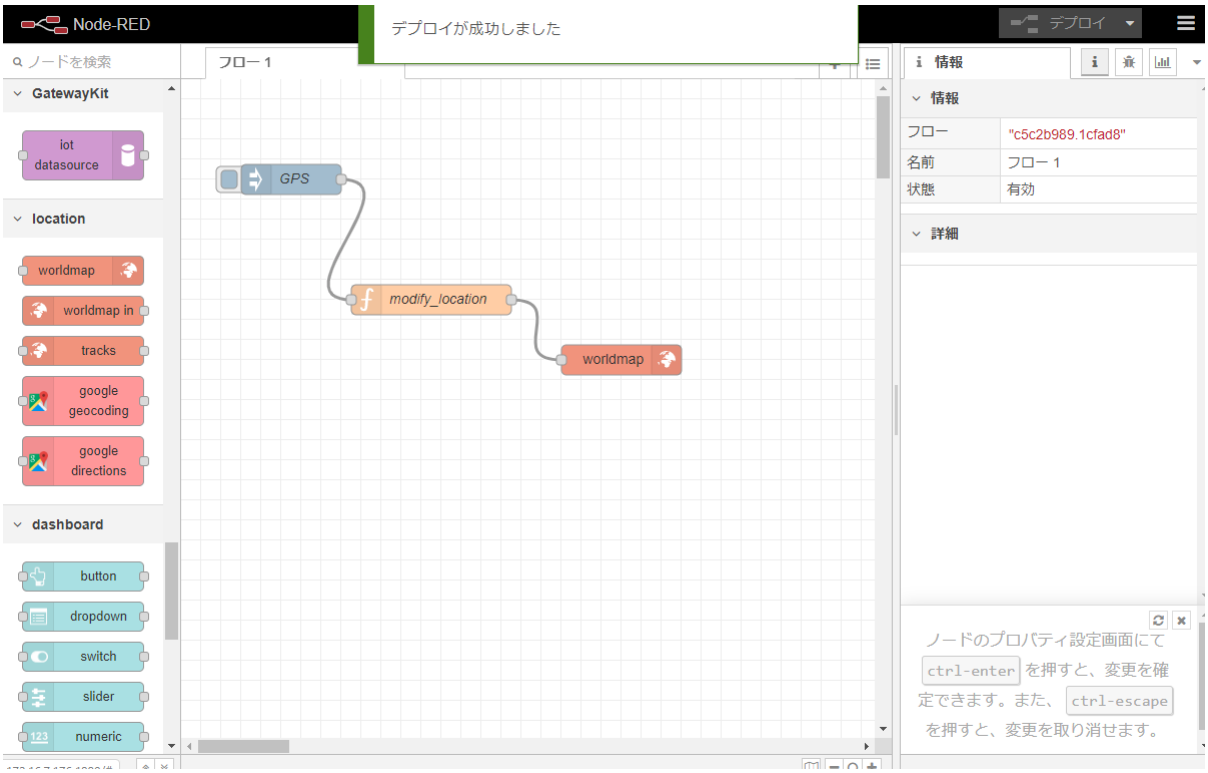
6, location ノードから worldmap ノードをドラッグしてシートにドロップします。



7, 配置した Inject ノード - function ノード、function ノード - worldmap ノード間を接続します。



8, Deploy ボタンをクリックします。

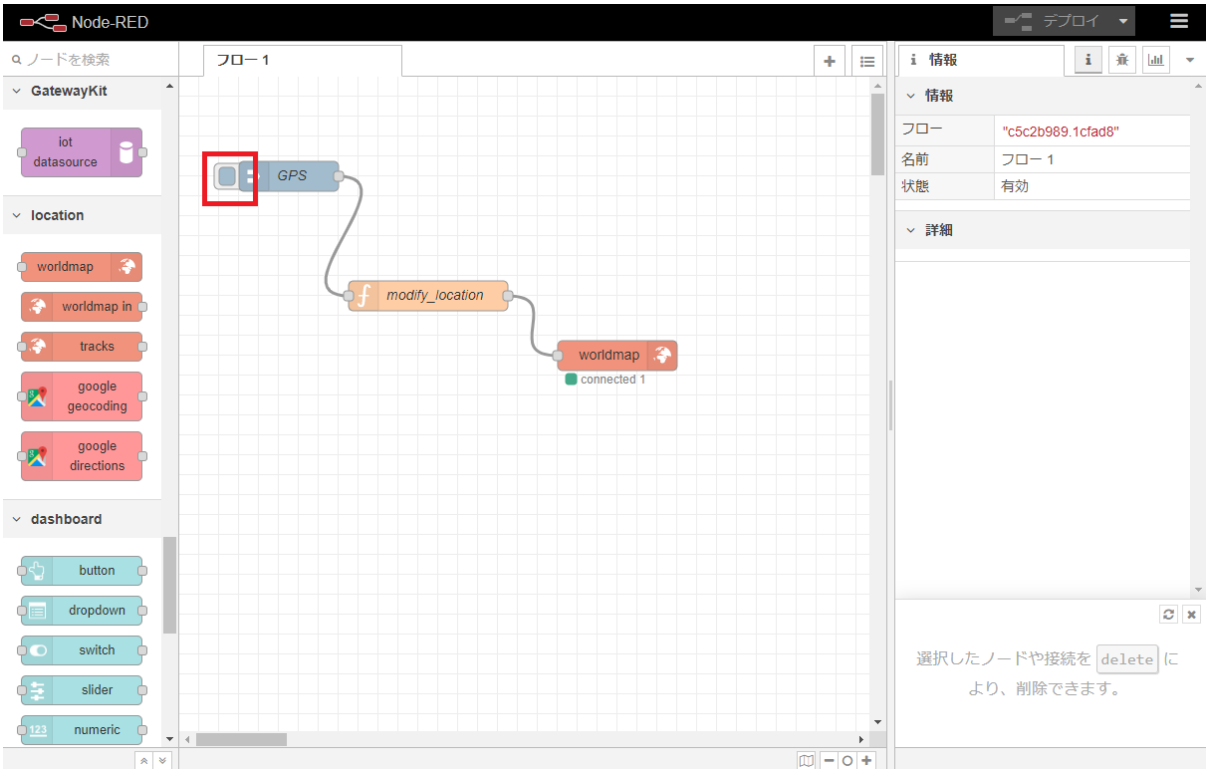


Deploy が完了すると、Deploy ボタンの背景が赤から黒に変わります。

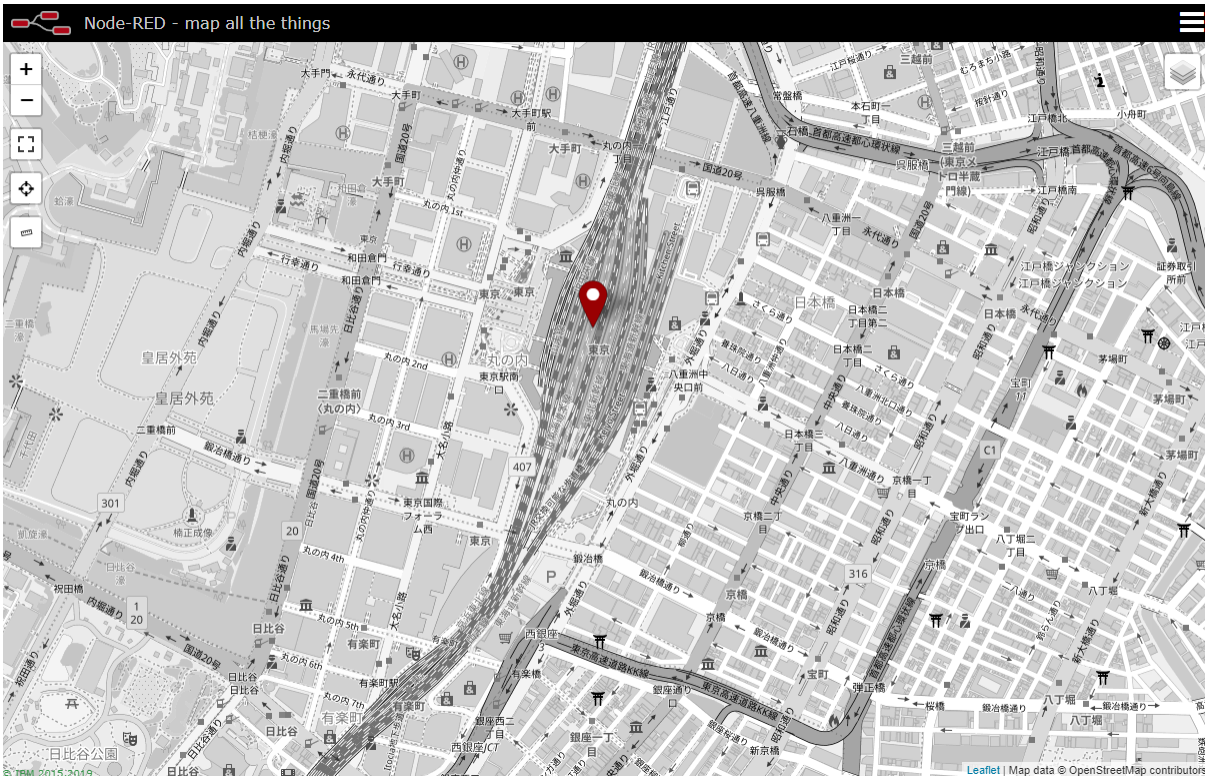
9, ブラウザの別タブにて以下にアクセスを行います。

http://192.168.254.254:1880/worldmap/

10, Inject ノードの左部のボタンを押します。



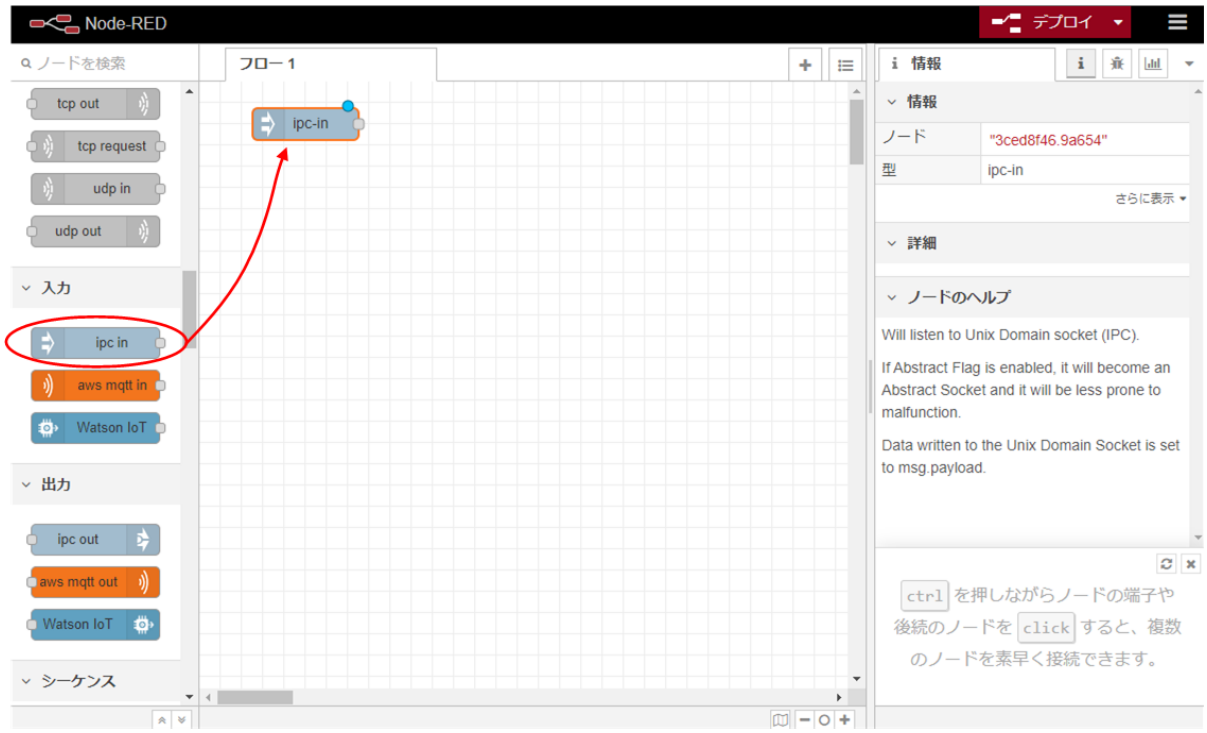
11, 別タブにて表示した地図の日本の東京駅にプロットされます。



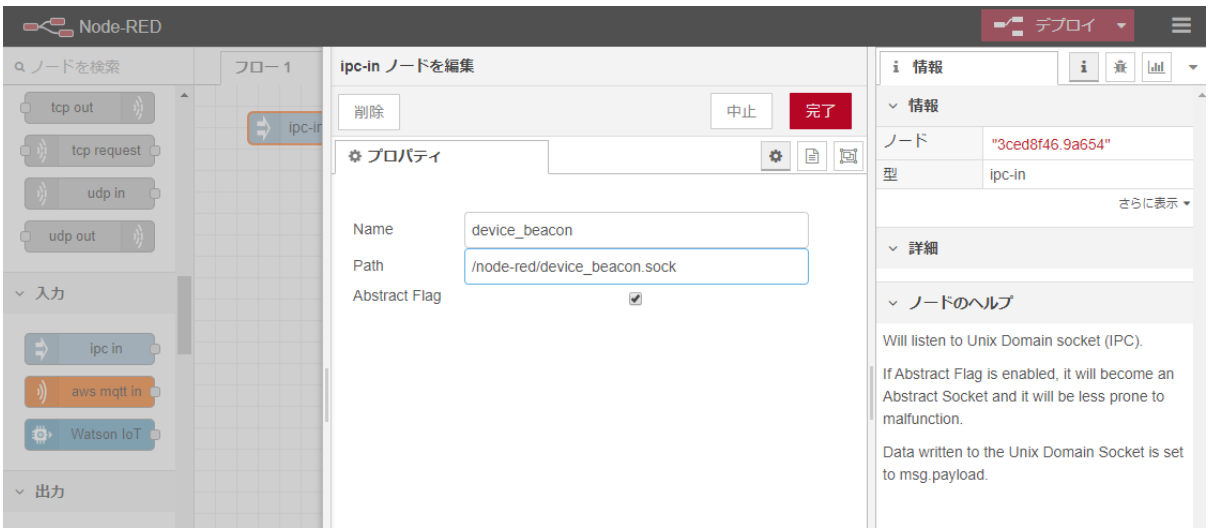
26.2 ビーコンデータへの位置情報付与

本項では IoT データ制御機能にて収集したビーコンデータに対して、LTE モジュール (NTT ドコモ/KDDI) または BWA モジュールを用いて取得した GPS 情報を付与するサンプルとなります。

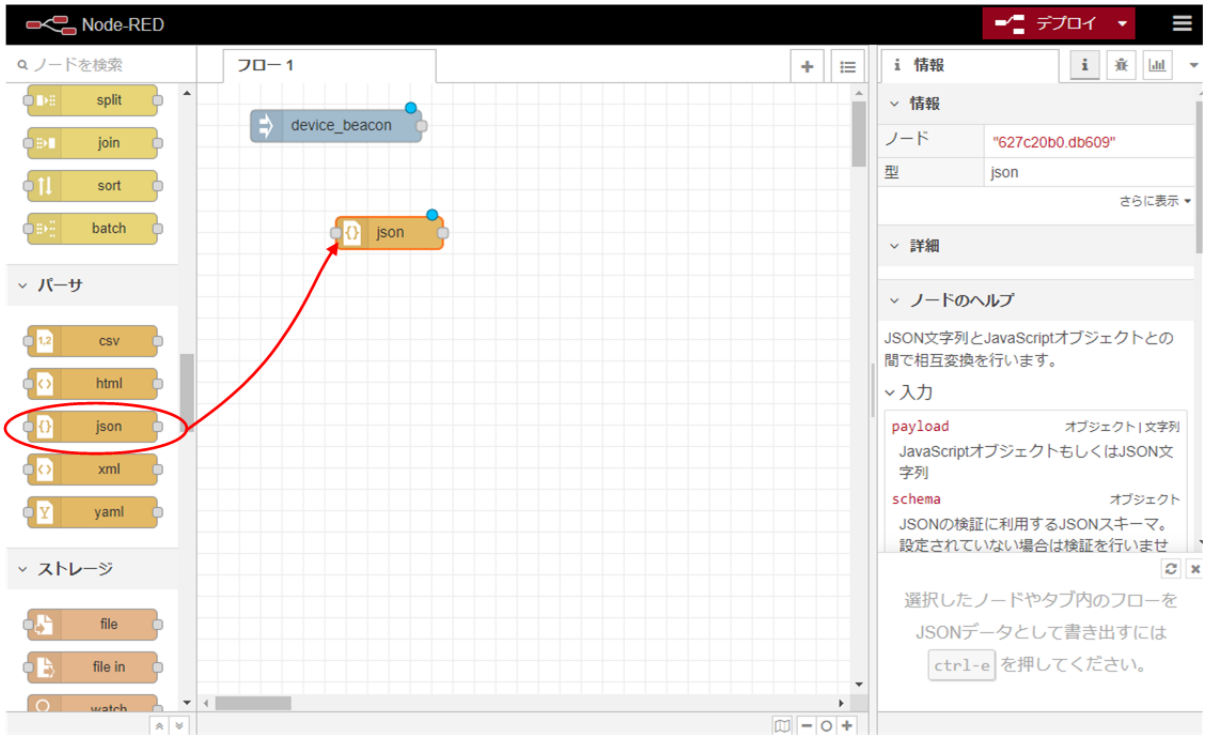
1, 入力ノードパレットから ipc-in ノードをドラッグしてシートにドロップします。



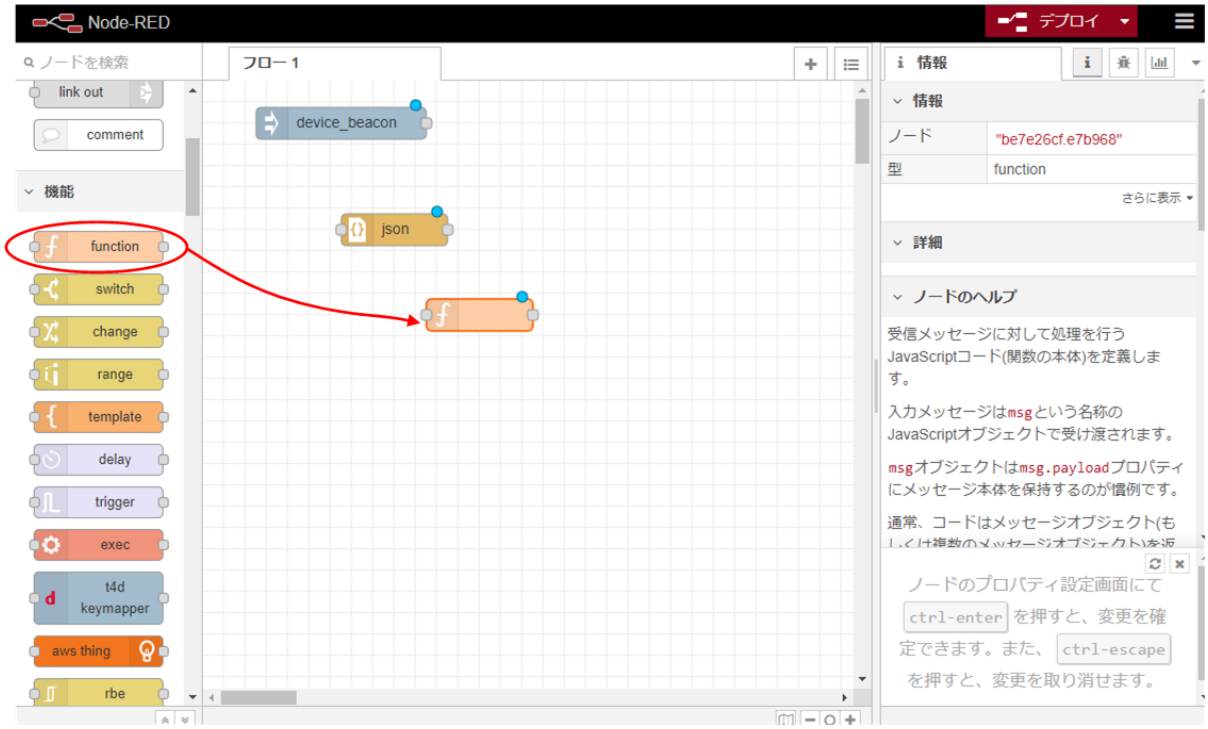
2, 配置した ipc ノードをダブルクリックし、ノードの編集を行います。Name に"device_beacon"と入力し、Path に"/node-red/device_beacon.sock"と入力します。尚、Abstract Flag をチェックしたままとします。入力完了後、完了ボタンを押します。



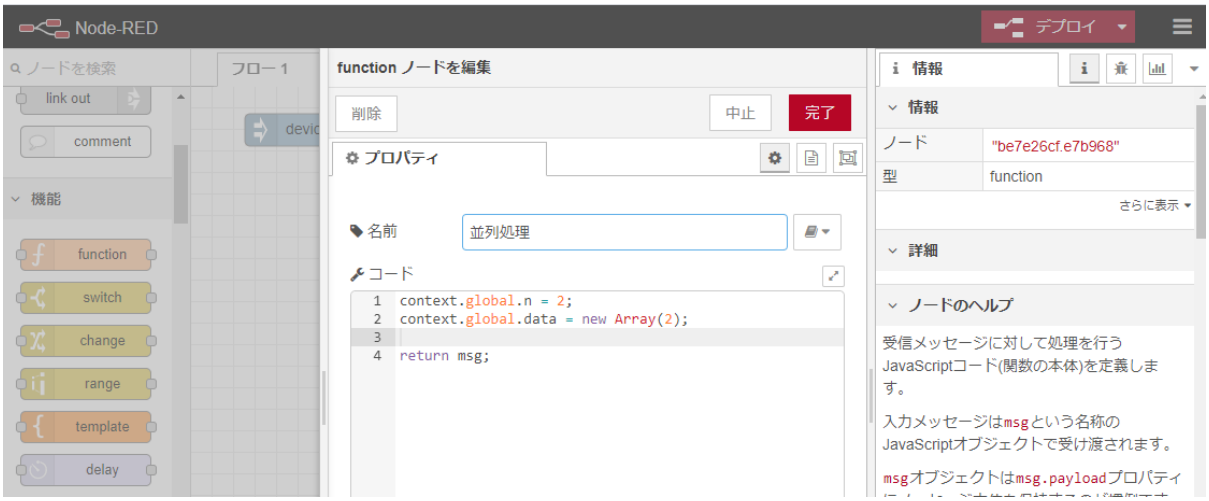
3, パーサノードパレットから json ノードをドラッグしてシートにドロップします。



4, 機能ノードパレットから function ノードをドラッグしてシートにドロップします。



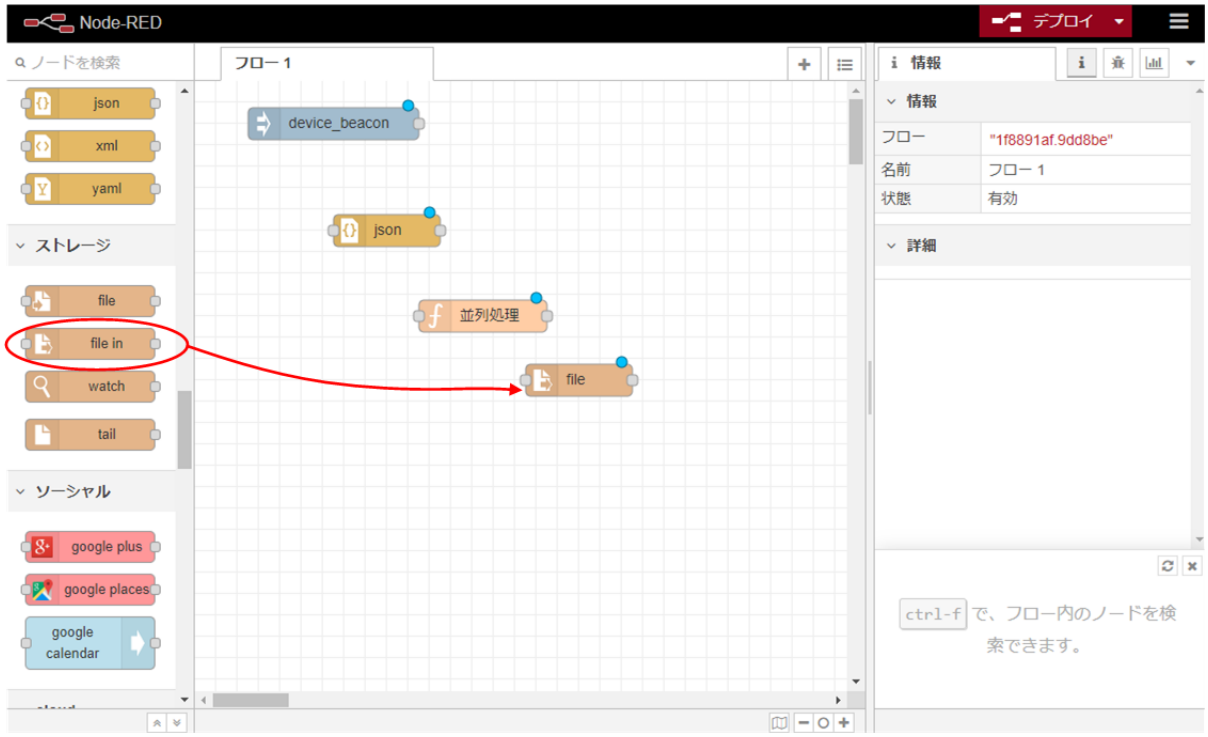
5, 配置した function ノードをダブルクリックし、ノードの編集を行います。名前はビーコンデータと GPS 情報の 2 種類を扱う為、"並列設定"とします。後述のコードを入力後、完了ボタンを押します。



また、コードは以下を設定します。

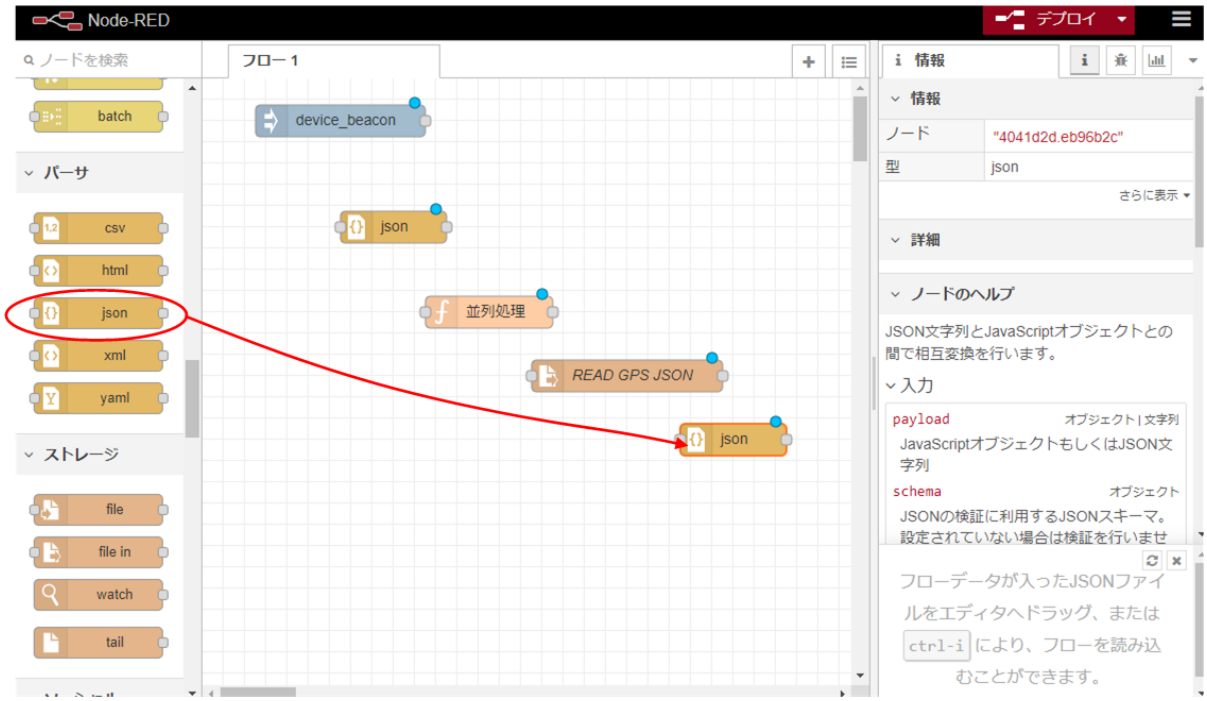
```
context.global.n = 2;
context.global.data = new Array(2);
return msg;
```

6, ストレージノードパレットから file in ノードをドラッグしてシートにドロップします。

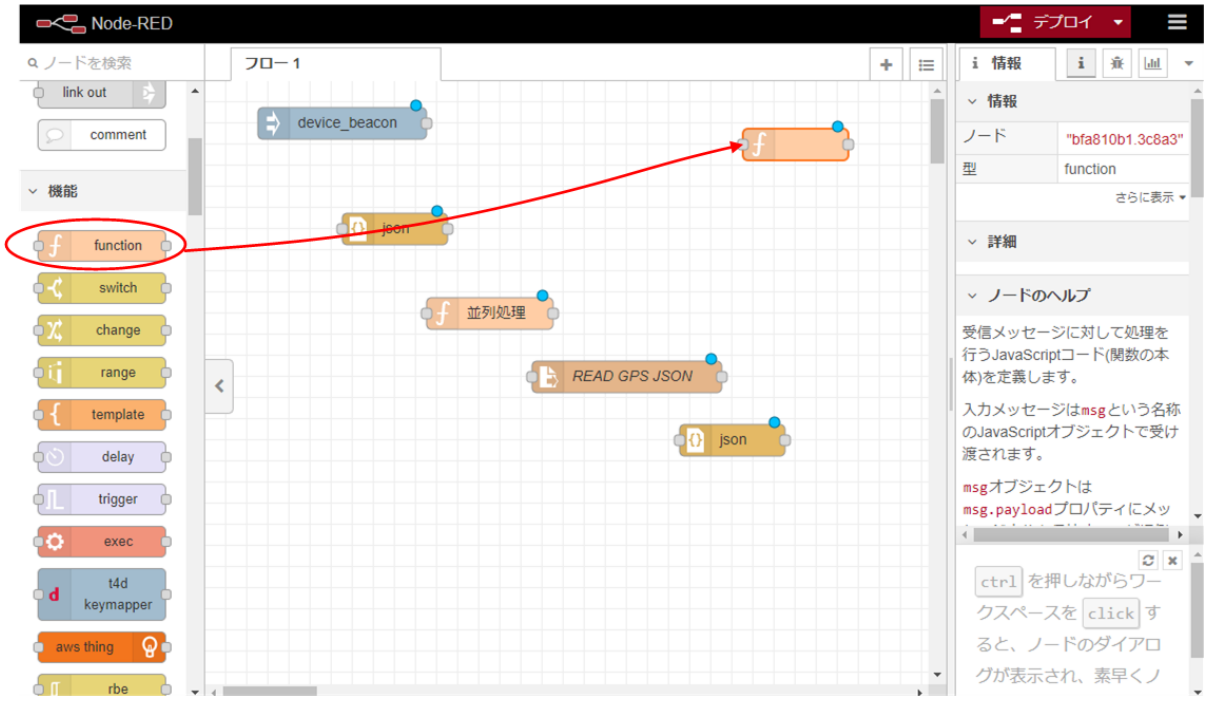


7, 配置した file in ノードをダブルクリックし、ノードの編集を行います。 ファイル名は"/tmp/.gps_posi.json"とします。また、名前は"READ GPS JSON"とします。尚、出力形式は文字列からの変更はありません。各項目入力後、完了ボタンを押します。

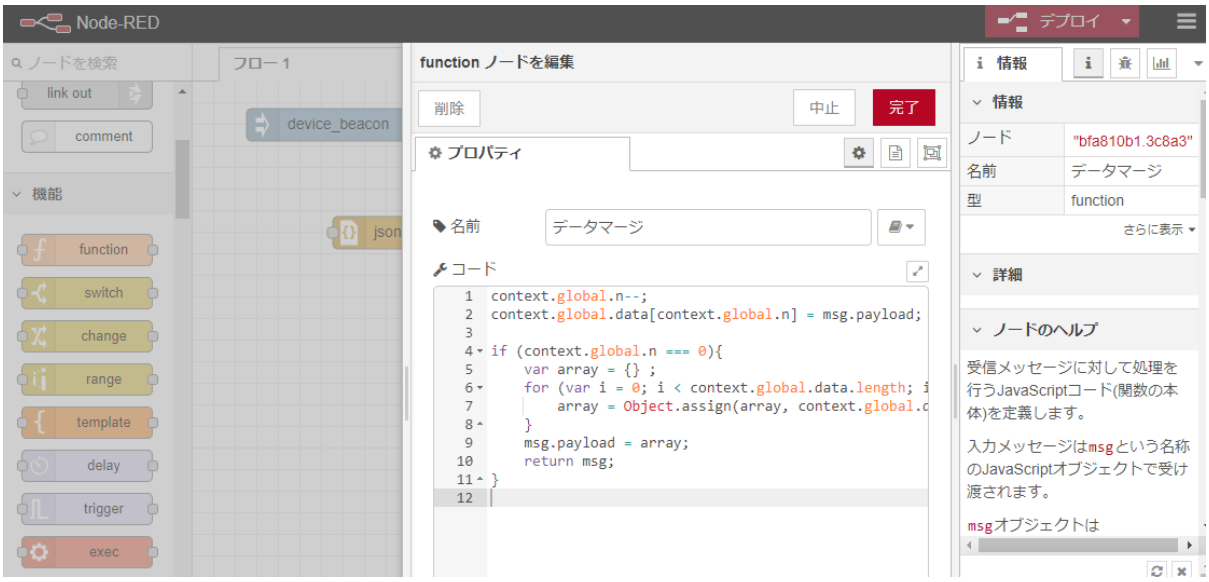
8, パーサノードパレットから json ノードをドラッグしてシートにドロップします。



9, 機能ノードパレットから function ノードをドラッグしてシートにドロップします。



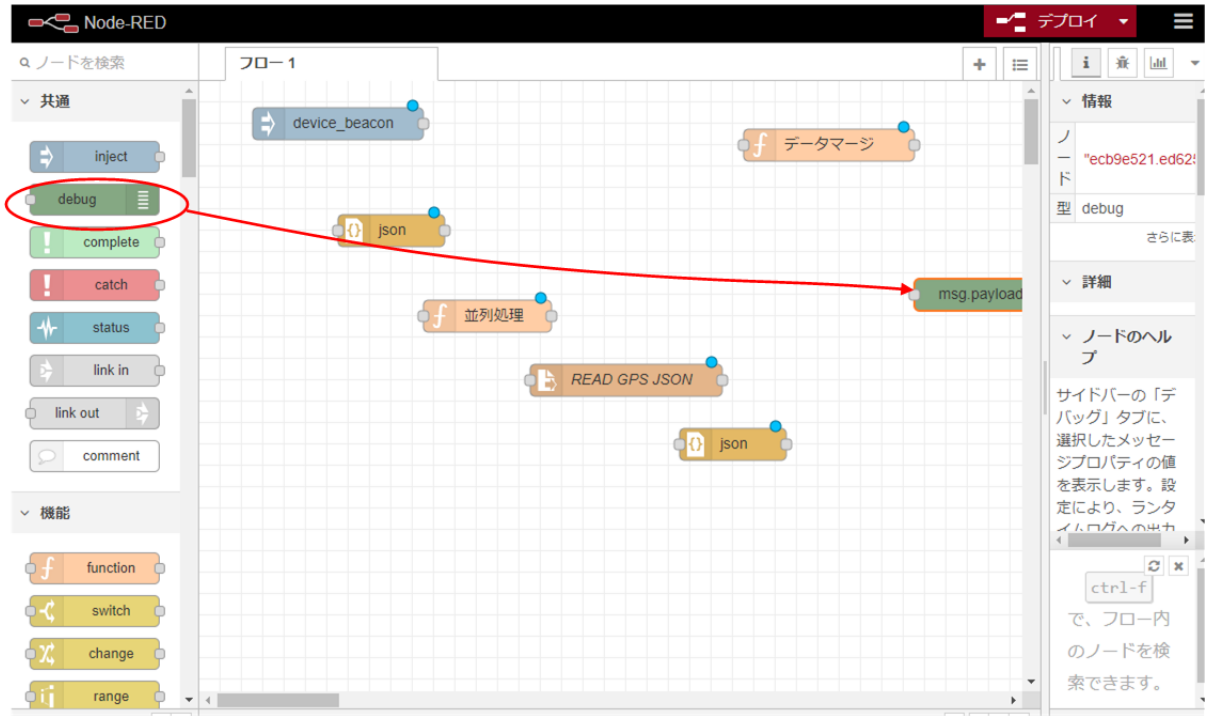
10, 配置した function ノードをダブルクリックし、ノードの編集を行います。名前はビーコンデータと GPS 情報の 2 種類のデータをマージする為、"データマージ"とします。後述のコードを入力後、完了ボタンを押します。



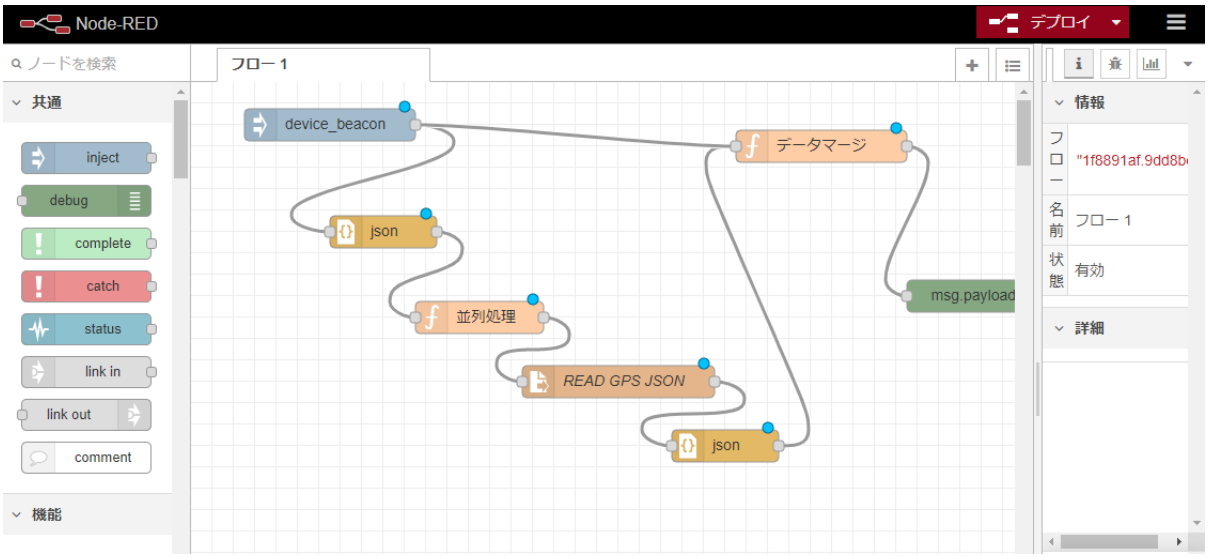
また、コードは以下を設定します。

```
context.global.n--;
context.global.data[context.global.n] = msg.payload;
if (context.global.n === 0){
        var array = { } ;
        for (var i = 0; i < context.global.data.length; i++) {
                array = Object.assign(array, context.global.data[i]);
        }
        msg.payload = array;
        return msg;
}
```

11, 出力結果を確認する為、出力ノードパレットから debug ノードをドラッグしてシートにドロップします。



12, 以下の図のように各ノード間を接続します。接続完了後、デプロイボタンを押します。これにより、ビーコンデータを Node-RED 側で受信した際に GPS 情報ファイルのデータをマージし、debug ノードにマージした情報が出力されます。

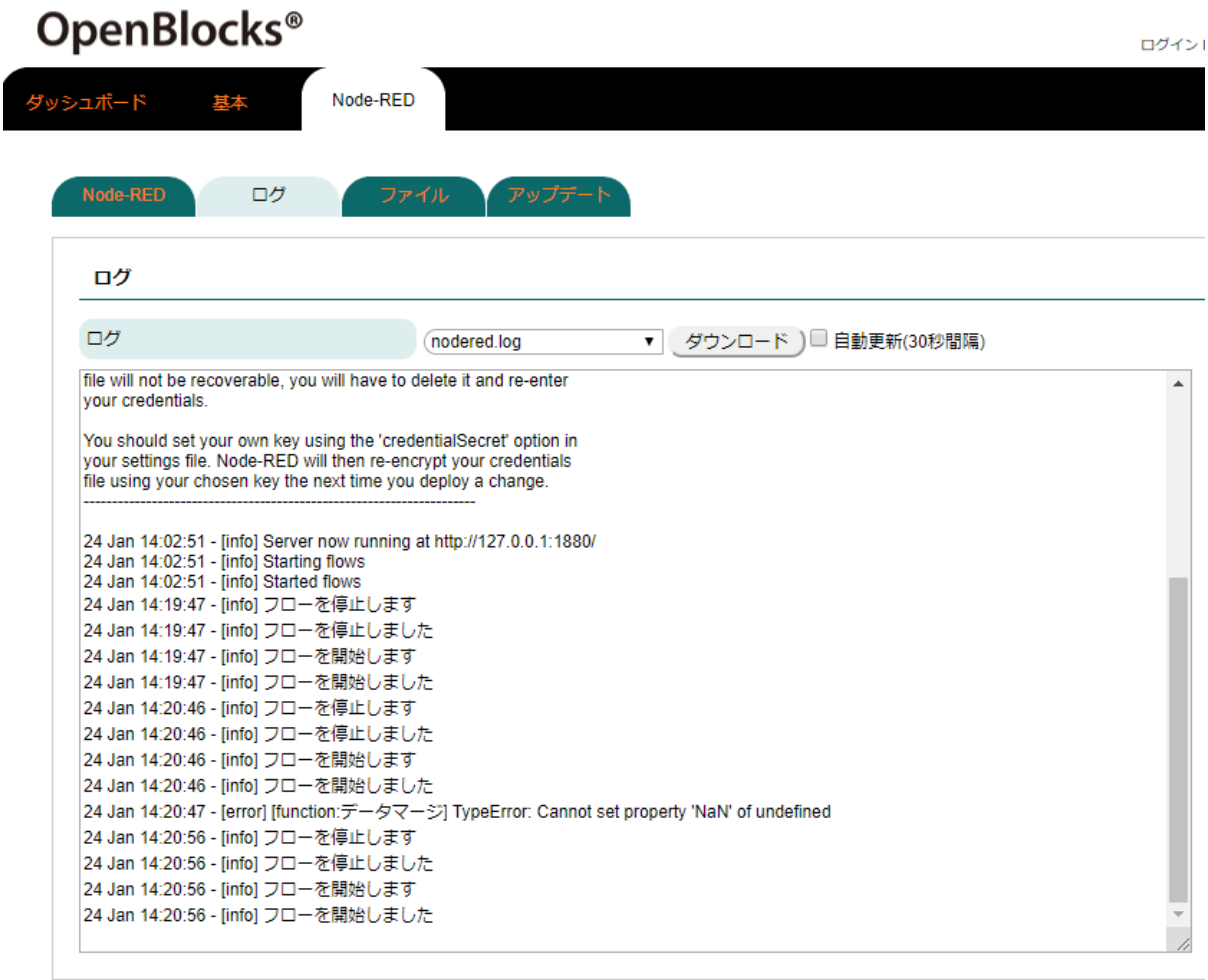


第27章　その他

27.1 Node-RED ログについて

Node-RED 自体のログについては、「Node-RED」→「ログ」タブから確認が行えます。

閲覧したいログファイルをログ欄から選択します。



ログファイルの選択を行うと、対象ログファイルの末尾が表示されます。

また、ダウンロードボタンを押すことによりログファイルのダウンロードが行えます。

27.2 Node-RED へのノード追加

Node-RED に対して対応するファイルをアップロードすることにより、Node-RED に対してノードを追加することができます。

ノード用のファイルアップロードは「Node-RED」→「ファイル」から行えます。尚、本機能でアップロードする箇所の機能については、多言語等の対応機能は含まれておりません。

node ファイル	
項目	備考
アップロード	Node-RED のアップロード用に単体のファイルをアップロードが行えます。 アップロードするファイルを選択後、対応するアップロードボタンを押してください。
一括アップロード	Node-RED のアップロード用に複数のファイルをアップロードが行えます。複数のファイルをアップロードする場合、tar 形式に圧縮したファイルを指定してください。 アップロードするファイルを選択後、対応するアップロードボタンを押してください。

アップロードしたノードファイルを Node-RED に反映させる場合には、Node-RED 自体の再起動が必要となります。後述のダッシュボードからプロセスの再起動を推奨します。

Node-RED のノードの作成方法については以下を参照してください。

https://nodered.jp/docs/creating-nodes/first-node

また、本項によるノード追加の場合、ノードの公開は行われないため package.json の作成等は不要です。

尚、フォーマットエラー等のファイル (html 及び js ファイル) をアップロードした場合、読み込みがスキップされ、対象ノードは表示されません。

27.3 Node-RED の Config 編集

「Node-RED」タブにて使用 Config 設定にてユーザー定義コンフィグを選択している場合、「編集」タブが表示されます。

この「編集」タブから Node-RED 自体のコンフィグ情報を編集できます。

編集	
項目	備考
編集対象ファイル	"setting.js"を選択してください。 ファイルを選択した段階で、現状使用するコンフィグファイルの内容がロードされます。 最終保存時のユーザーコンフィグを読み込みたい場合は、「Load(ユーザーコンフィグ)」ボタンを押してください。 また、システム作成のコンフィグを読み込みたい場合は、「Load(システムコンフィグ)」ボタンを押してください。 ※本項目の編集については十二分に理解がある方を前提としています。

コンフィグファイルの編集後、保存ボタンを押すことにより反映されます。

また、編集中のコンフィグファイルの内容をエクスポートボタンを押すことで保存することが可能です。

27.4 Node-RED のプロセス状況について

Node-RED を使用設定にしている場合、ダッシュボードにて Node-RED 自体のプロセス状況を確認することができます。

プロセス状況に応じて、「起動」・「停止」・「初期化」ボタン等が表示されます。

「初期化」ボタンは Node-RED のフロー状況等が削除されますので、ご注意ください。

27.5 Node-RED の HTTPS 化

Node-RED のダッシュボードへのアクセスは通常では HTTP 通信となります。

セキュリティ強化のため HTTPS 通信へと変更を行いたい場合には、以下の方法を実施することで切り替え可能となります。

※ HTTPS 通信に切り替えた場合、WEB UI の Node-RED タブ内のリンクボタンからのリンクは HTTP 前提となっている為、リンク切れが発生します。

1, 証明書の作成

本項では/var/tmp ディレクトリ配下に自己署名証明書を作成しています。正式なものとして使用するには推奨いたしません。

```
# cd /var/tmp
# openssl genrsa 2048 > privatekey.pem
# openssl req -new -key privatekey.pem 2> /dev/null > server.csr <<!
JP
Tokyo
＜空行＞
＜空行＞
＜空行＞
＜空行＞
＜空行＞
＜空行＞
＜空行＞
!
# openssl x509 -days 365 -req -signkey privatekey.pem < server.csr 2> /dev/null > certificate.pem
```

※＜空行＞は何も入力しないサンプルとなっております。実際には、各項目にあったパラメータを入力してください。

2, WEB UI の Node-RED タブ内の使用 Config 設定をユーザー定義コンフィグに設定し保存します。

3, Node-RED カテゴリの編集タブにてコンフィグを修正します。 編集対象ファイルを setting.js を選択し、テキストボックス内を変更します。 変更内容は ”var fs ”部のコメント解除及び https ディレクティブのコメント解除・パス修正となります。

変更前

……
// The *https* setting requires the *fs* module. Uncomment the following
// to make it available:
//var fs = require("fs");
……
// The following property can be used to enable HTTPS
// See http://nodejs.org/api/https.html#https_https_createserver_options_requestlistener
// for details on its contents.
// See the comment at the top of this file on how to load the *fs* module used by
// this setting.
//
//https: {
// key: fs.readFileSync('privatekey.pem'),
// cert: fs.readFileSync('certificate.pem')
//},
……

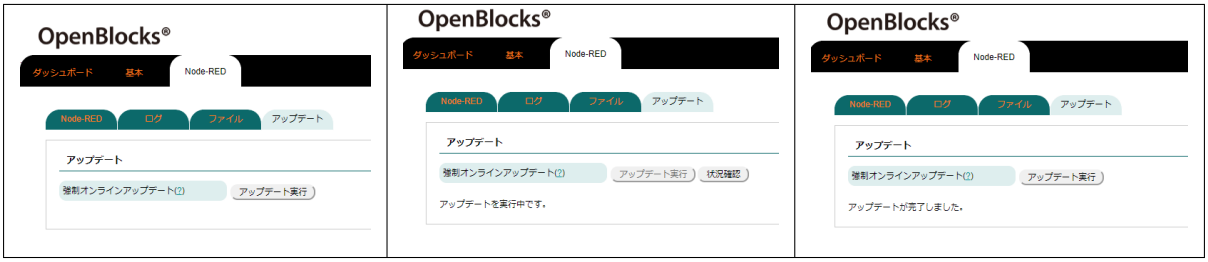
変更後

……
// The *https* setting requires the *fs* module. Uncomment the following
// to make it available:
var fs = require("fs");
……
// The following property can be used to enable HTTPS
// See http://nodejs.org/api/https.html#https_https_createserver_options_requestlistener
// for details on its contents.
// See the comment at the top of this file on how to load the *fs* module used by
// this setting.
//
https: {
key: fs.readFileSync('/var/tmp/privatekey.pem'),
cert: fs.readFileSync('/var/tmp/certificate.pem')
},
……

5, 変更完了後、保存ボタンを押すことで HTTPS 化は完了となります。

27.6 Node-RED のバージョンアップ

インターネット接続環境の場合、Node-RED 自体のアップデートを行う事が可能です。弊社で用意している Node-RED 及び Node-RED のモジュールのアップデートは、「Node-RED」→「アップデート」タブから確認が行えます。



アップデート	
項目	備考
強制オンラインアップデート	アップデート実行ボタンを押すことにより、オンラインアップデートが行われます。 アップデート処理中は状況確認ボタンにて、アップデートログが別ウィンドウにて表示可能です。 また、アップデートが完了すると完了の旨のポップアップが表示されますので、OK ボタンを押してください。 本機能はオンライン環境である必要がありますので、ご注意ください。また、Node-RED 及び Node- RED モジュールについては各コントリビューターが用意しているものであるため、互換性が担保されているものではありません。そのため、事前に確認機等にてアップデート及び動作確認を推奨いたします。

第VII部

Samba設定ガイド

第28章　はじめに

本書は、OpenBlocks シリーズに搭載可能なファイル共有ソフト (Samba) の使用方法を解説しています。

第29章　Samba設定

29.1 Sambaのインストール

本製品出荷時では、Samba はインストールされておりません。そのため、WEB UI の「メンテナンス」→「機能拡張」タブから Samba をインストールしてください。

WEB UI の「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

インストール機能のリストから「Samba」を選択します。

その後、インストールの「実行」ボタンを押し、インストールを行ってください。

尚、インストール完了後には反映を行うため、本体再起動が必要となります。そのため、「メンテナンス」

→「停止・再起動」から本製品の再起動を行ってください。

29.2 Samba 起動設定

Samba のインストールが完了している場合、サービス WEB UI の「拡張」→「Samba」タブに Samba の項目が表示されます。

使用設定を「使用する」に設定し、保存することでファイル共有用サーバーとして扱うことができます。

アクセス用に Samba 用ユーザーとして"obssamba"を用意しております。デフォルトパスワードは"P1atDeb1an"です。セキュリティの観点上、パスワードは変更してください。

Samba	
項目	備考
使用設定	Samba の使用設定を行います。Samba を使用する場合には、「使用する」を選択してください。
Samba 用ホスト名	同一ネットワーク内の WindowsPC 等で表示する名前を設定します。
Samba 用ディレクトリパス	ファイル共有として扱うディレクトリパスを設定します。
認証設定	ファイル共有先にアクセスする場合の、セキュリティの設定を行います。 認証なし：共有先にアクセスする場合、認証は発生しません。 Samba ユーザーのみ：共有先にアクセスする場合、Samba 用のユーザーのみアクセスできます。 ※ Samba 用ユーザーとして obssamba を用意しています。
パスワード	入力フォームに対して、パスワードを設定し変更ボタンを押すことで、Samba 用ユーザー (obssamba) のパスワードが変更可能です。

設定完了後、保存ボタンを押すことにより Samba が起動・停止します。

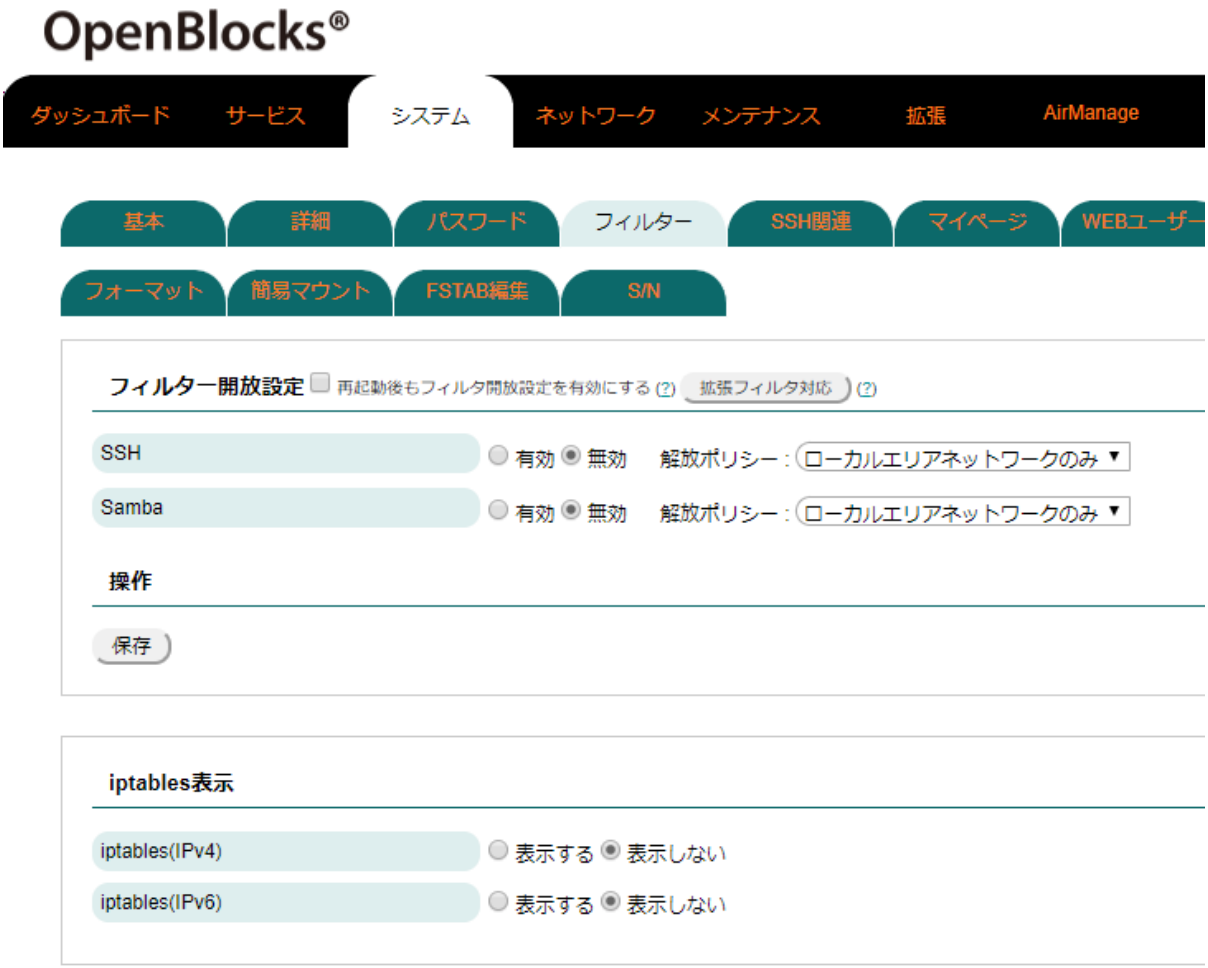
尚、Samba 機能を使用する状態にて、保存ボタンを押した場合にて、フィルターが解放されていない場合には、解放確認のポップアップが表示されます。また、解放ポリシー設定ポップアップが出ますので、設定したいポリシーのボタンを押してください。

尚、本機能での書き込み等についてはストレージ寿命を縮めることが想定されます。

そのため、ディレクトリパス先については外部ストレージや tmpfs 等の本体ストレージに影響のない場所を推奨いたします。

29.3 Samba パケットフィルタについて

Samba のインストールが完了している場合、ファイル共有用のプロトコル通信の為、Samba のフィルターを開放する必要があります。そのため、WEB UI の「システム」→「フィルター」タブにてフィルターを開放してください。



デフォルトでは Samba 用のアクセスはできないようにフィルターが適用されています。

"有効"に設定し、保存ボタンを押してください。

第30章　その他

30.1 外部ストレージとの連携について

WEB UI の標準機能として、外部ストレージを自動的にマウントする機能が存在しています。ファイルサーバとして扱う場合、一定以上の書き込みが想定されます。そのため、Samba によるファイル共有用の書き込み先については、外部ストレージマウント機能と連携し運用する事を推奨します。

30.2 Samba 起動設定

ファイル共有において書き込み一定以上行い、対象ストレージの使用率が一定以上となるケースが想定されます。WEB UI の標準機能として、ディスクの使用率が一定以上となった場合、特定ディレクトリ配下の一定期間以上前のファイル削除を行うことが可能です。そのため、ストレージクリーンナップ機能と連携し運用する事を推奨します。

第VIII部

セキュリティガイド

第31章　はじめに

本書は、OpenBlocks シリーズに搭載可能な WEB UI 込みでの一部不正アクセス対応のセキュリティ設定の使用方法を解説しています。

第32章　セキュリティ設定

32.1 セキュリティ機能のインストール

本製品出荷時では、セキュリティ設定 WEB UI はインストールされておりません。そのため、WEB UI の「メンテナンス」→「機能拡張」タブからセキュリティ設定用 WEB UI のインストールを行います。

WEB UI の「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

WEB UI の「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

インストール機能のリストから「セキュリティ」を選択します。

その後、インストールの「実行」ボタンを押し、インストールを行ってください。

尚、インストール完了後には反映を行うため、本体再起動が必要となります。そのため、「メンテナンス」→「停止・再起動」から本製品の再起動を行ってください。

32.2 セキュリティ使用設定について

セキュリティ機能のインストールが完了している場合、サービス WEB UI の「拡張」→「セキュリティ」タブにセキュリティの設定項目が表示されます。

使用設定を「使用する」に設定し、適用するセキュリティ使用を「使用する」にし保存することで対象のセキュリティ機能が適用されます。

尚、本機能は起点の初回不正攻撃から、一定時間の間に指定不正攻撃回数 (初回含む) 分の不正攻撃があった場合、対象 IP アドレスの対象サービスへのアクセスを拒否します。

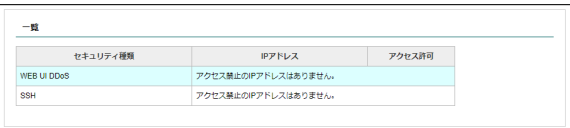
セキュリティ	
項目	備考
使用設定	セキュリティ機能の使用設定を行います。セキュリティ機能を使用する場合には、「使用する」を選択してください。
ログレベル	出力するログレベルを以下から選択します。 ・INFO ・CRITICAL ・ERROR ・WARNING ・NOTICE ・DEBUG 尚、基本的には「INFO」から変更す必要はありません。
除外ネットワークアドレス	セキュリティ機能から除外するネットワークアドレスを指定します。尚、複数指定する場合には、空白にて追加してください。 Ex.) 192.168.254.0 のネットワークを追加する場合 "127.0.0.1/8 192.168.254.0/24"
不正対象間隔 [sec]	対象 IP アドレスから対象サービスへの不正攻撃が開始された際に、同様の不正攻撃としてグループ化する時間を指定します。
拒否時間 [sec]	対象 IP アドレスから対象サービスへのアクセスを拒否する時間を指定します。
不正攻撃回数	不正対象間隔内に不正攻撃が一定回数来た場合、アクセス拒否と判定する際の回数を指定します。
SSH 使用設定	SSH のログイン失敗及び DDoS 攻撃によるセキュリティ機能の使用設定を行います。 使用する場合には、「使用する」を選択して下さい。
WEB UI DDoS 使用設定	WEB UI 使用ポートでの HTTP/HTTPS アクセスにおける、403/404 アクセス攻撃によるセキュリティ機能の使用設定を行います。 使用する場合には、「使用する」を選択して下さい。

各使用設定を「使用する」に保存した場合、セキュリティ機能が有効となります。

32.3 不正攻撃のアクセス拒否解除

セキュリティ機能を有効とした場合、「拡張」→「セキュリティ」タブのページ下部に有効となっているセキュリティ一覧が表示されます。

また、アクセス拒否となっている IP アドレス/対象サービスの組み合わせが表示されます。

アクセス拒否が存在しない場合	SSH にアクセス拒否が存在する場合
	

一覧

セキュリティ種類	IPアドレス	アクセス許可
WEB UI DDoS	アクセス禁止のIPアドレスはありません。	
SSH	アクセス禁止のIPアドレスはありません。	

一覧

セキュリティ種類	IPアドレス	アクセス許可
WEB UI DDoS	アクセス禁止のIPアドレスはありません。	
SSH	172.16.7.153	アクセス許可

不正攻撃等が無く、正常に稼働している場合には左図のようにアクセス拒否となっている IP アドレスは表示されません。

不正攻撃が行われ、アクセス拒否となっている IP アドレスが存在する場合、左図のように IP アドレス及び「アクセス許可」ボタンが表示されます。

尚、「アクセス許可」ボタンは対象 IP アドレスから対象サービスへの再アクセスを許容するようにします。問題のないユーザーがログイン失敗等による再アクセスを行わせたい場合にご使用ください。

第33章　その他

33.1 機能拡張等について

本機能は fail2ban デーモンを使用しています。お客様ご自身でセキュリティ機能を拡張したい場合には以下のページをご確認ください。尚、拡張によるファイル編集及び追加・削除についてはお客様責任となりますのでご注意ください。

https://www.fail2ban.org/wiki/index.php

WEB UI の拡張タブ内の保存ボタンイベントにて、以下のファイルは再生成されます。

・/etc/fail2ban/fail2ban.conf

・/etc/fail2ban/jail.local

拡張の際にこれらのファイルについて上書きされたくない場合には、以下のファイルを用意してください。

上書禁止ロック用ファイル
/var/webui/config/fail2ban.userlock

また、既存の/etc/fail2ban/配下のファイルを編集した場合、元に戻すことはできません。もとに戻したい場合には、ファクトリーリセット及び再インストールを実施してください。

第IX部

カメラ設定ガイド

第34章　はじめに

本書は、OpenBlocks シリーズに搭載可能な USB 及び WEB カメラを用いたストリーミング機能及び動体検知の使用方法を解説しています。

第35章　カメラ機能

35.1 カメラ機能のインストール

本製品出荷時では、カメラ機能はインストールされておりません。そのため、WEB UI の「メンテナンス」→「機能拡張」タブからカメラ機能をインストールしてください。

WEB UI の「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

WEB UI の「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

インストール機能のリストから「カメラ」を選択します。

その後、インストールの「実行」ボタンを押し、インストールを行ってください。

尚、インストール完了後には反映を行うため、本体再起動が必要となります。そのため、「メンテナンス」→「停止・再起動」から本製品の再起動を行ってください。

35.2 カメラ機能設定

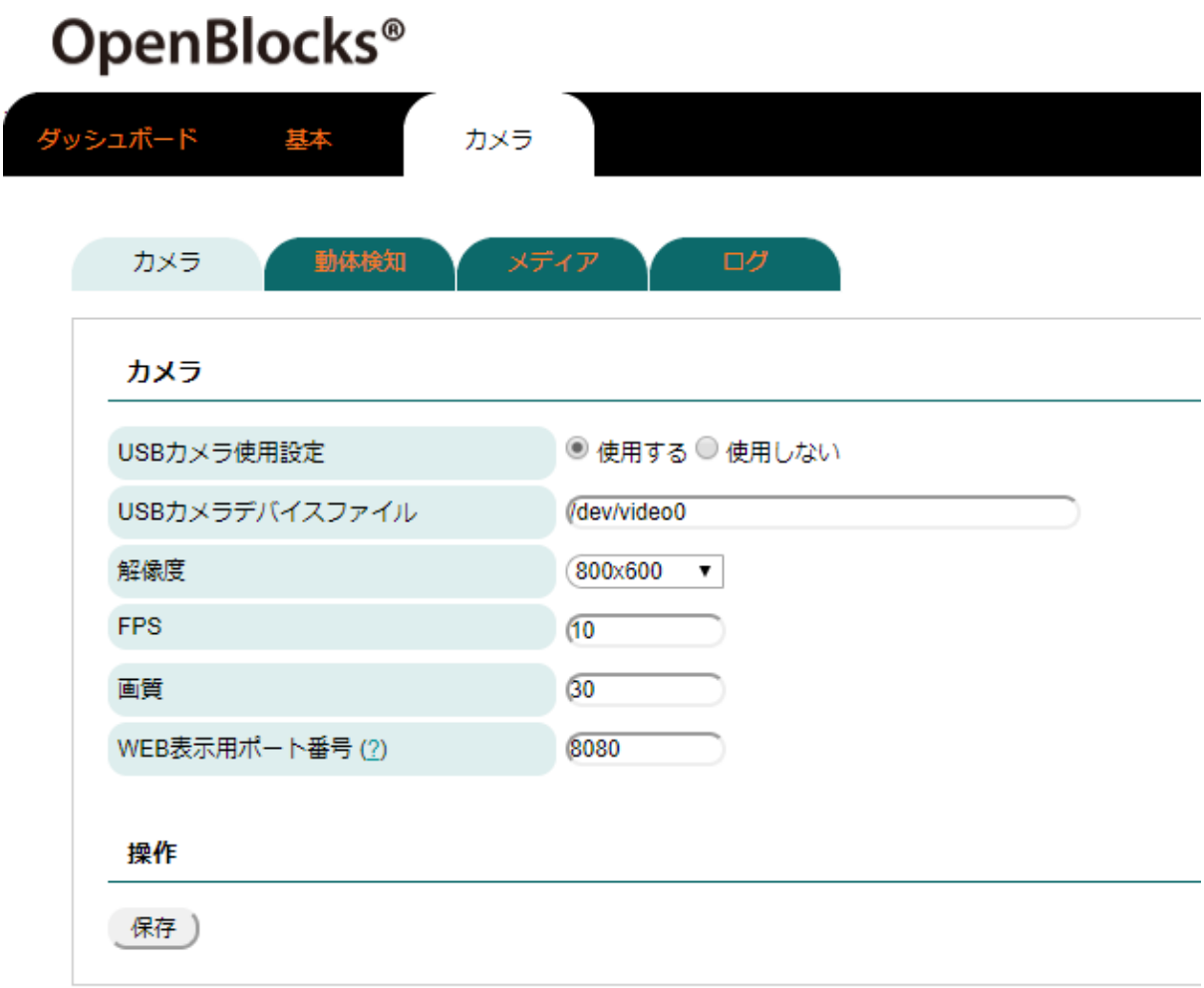
カメラ機能のインストールが完了している場合、サービス WEB UI の「サービス」→「基本」タブにカメラの項目が表示されます。

このカメラのリンクを押して下さい。これにより、「カメラ」タブへと遷移します。使用したい機能に関しては、後述の内容を参照の上適宜設定を行ってください。

35.3 カメラ ストリーミング機能

「カメラ」→「カメラ」タブでは OpenBlocks 本体に接続しているカメラの画像データを WEB UI 上にストリーミングとして表示することができます。また、WEB UI にてストリーミングの表示部の確認を行おうとした場合、URI プロキシ機能を用います。

接続しているカメラ動作の確認を行う場合に、ご使用ください。



カメラ	
項目	備考
USB カメラ設定	USB カメラでのストリーミング表示を使用する場合には、「使用する」を選択してください。
USB カメラデバイスファイル	OpenBlocks がカメラとして認識しているデバイスファイルを指定します。通常では変更する必要はありません。
解像度	カメラが撮影する際の解像度を選択します。 ※本リスト内に表示される解像度の一覧はカメラ自体とは連動しておりません。
FPS	フレームレートを設定します。ドキュメント作成時点での標準的な USB カメラの FPS は 30fps 程度ですので、1〜30 までの間で設定してください。
画質	撮影する際の画質を 1〜100 の間で設定します。(100 に近づくほど高画質となります) 画質が高い場合、CPU 負荷が高くなりますのでリソース状況確認の上、設定してください。
WEB 表示用ポート番号	WEB に表示するためのポート番号を指定します。本システムでは使用していない 8080 をデフォルトで設定しております。 ※ URI プロキシを用いて確認する場合には、このポート番号を設定して下さい。

設定完了後、保存ボタンを押すことによりカメラのストリーミング機能が起動・停止します。

使用する設定となっている場合、以下のように表示されます。

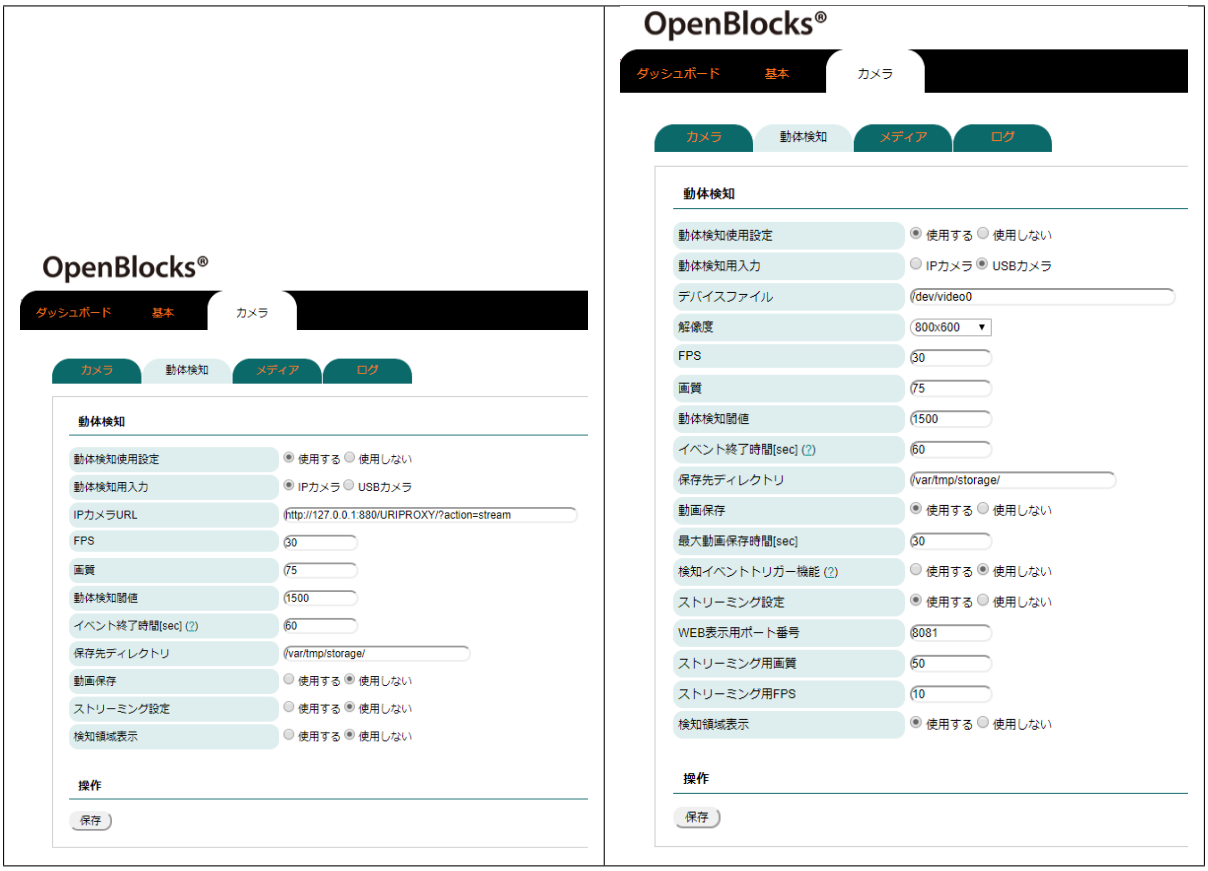
ストリーミング	
項目	備考
表示URI名	URI プロキシに設定したポート番号が存在する場合、自動入力されます。 また、「表示する」ボタンを押すことによりストリーミング画像が WEB UI 上に表示されます。
表示サイズ	WEB UI 上に表示されるストリーミング画像のサイズを選択できます。

※ WEB UI 上に表示されるストリーミング画像は 5 分間表示されます。

35.4 動体検知機能

「カメラ」→「動体検知」タブではストリーミングデータまたは OpenBlocks 本体に接続しているカメラを用いて動体検知を行うことができます。

動体を検知した際に動画の保存及び保存イベントにおけるスクリプトの実行処理についてもおこなうことができます。また、本機能についてもストリーミングを行うことができます。尚、ストリーミング機能については、URI プロキシ機能を用います。



動体検知	
項目	備考
動体検知使用設定	動体検知を使用する場合には、「使用する」を選択してください。
動体検知用入力	動体検知の検出として入力を「IP カメラ」または「USB カメラ」から選択します。
IP カメラ URL(IP カメラ使用時)	入力とする IP カメラの URL を指定します。前章のカメラストリーミング機能を入力とする場合は URI プロキシ機能を用いて以下のように設定してください。 http://127.0.0.1:880/<URI プロキシ名>/?action=stream また、外部のネットワークカメラを入力とすることも可能です。その場合、Basic 認証はサポートしておりますので、認証なしまたは Basic 認証でのアクセスを使用して下さい。
デバイスファイル(USB カメラ使用時)	OpenBlocks がカメラとして認識しているデバイスファイルを指定します。通常では変更する必要はありません。
解像度(USB カメラ使用時)	カメラが撮影する際の解像度を選択します。 ※本リスト内に表示される解像度の一覧はカメラ自体とは連動しておりません。
FPS	フレームレートを設定します。ドキュメント作成時点での標準的な USB カメラの FPS は 30fps 程度ですので、1〜30 までの間で設定してください。
画質	撮影する際の画質を 1〜100 の間で設定します。(100 に近づくほど高画質となります) 画質が高い場合、CPU 負荷が高くなりますのでリソース状況確認の上、設定してください。
動体検知閾値	動体と認識するフレーム間での差分ピクセルの閾値を設定します。解像度及び処理したい内容によって変更が必用となります。
イベント終了時間 [sec]	動体検出後、動きがなくなった場合に動体検知を終了とする時間を設定します。
保存先ディレクトリ	動体検知を行った際に動画を保存する際のディレクトリを指定します。 ※動体検知量によって書き込み量が変わる為、SD カード上を推奨いたします。
動画保存	動体を検知した際に、動画を保存を行うか設定します。

動体検知	
最大動画保存時間[sec](動画保存使用時)	動画を保存する際に最大で保存を行う動画の秒数を指定します。
検知イベントトリガー機能 (動画保存使用時)	動画保存を行った際に、スクリプトを実行するかどうかを指定します。 実行するスクリプトは「拡張」→「スクリプト編集」タブにて作成可能です。
ストリーミング設定	動体検知状況をストリーミング機能にて表示するかを設定します。使用する場合に「使用する」を選択してください。
WEB 表示用ポート (ストリーミング時)	WEB に表示するためのポート番号を指定します。本システムでは使用していない 8081 をデフォルトで設定しております。 ※ URI プロキシを用いて確認する場合には、このポート番号を設定して下さい。
ストリーミング用画質 (ストリーミング時)	ストリーミング時の画質を 1〜100 の間で設定します。(100 に近づくほど高画質となります)
ストリーミング用 FPS(ストリーミング時)	ストリーミング時のフレームレートを設定します。1〜100 まで設定は可能ですが、カメラの FPS を上回らない値を設定してください

設定完了後、保存ボタンを押すことにより動体検知機能が起動・停止します。

ストリーミング機能を使用する設定となっている場合、以下のように表示されます。

OpenBlocks®

ダッシュボード　基本　カメラ

カメラ　動体検知　メディア　ログ

動体検知

動体検知使用設定	◉ 使用する ○ 使用しない
動体検知用入力	◉ IPカメラ ○ USBカメラ
IPカメラURL	http://127.0.0.1:880/URIPROXY/?action=stream
FPS	30
画質	75
動体検知閾値	1500
イベント終了時間[sec] (?)	60
保存先ディレクトリ	/var/tmp/storage/
動画保存	◉ 使用する ○ 使用しない
最大動画保存時間[sec]	30
検知イベントトリガー機能 (?)	○ 使用する ◉ 使用しない
ストリーミング設定	◉ 使用する ○ 使用しない
WEB表示用ポート番号	8081
ストリーミング用画質	50
ストリーミング用FPS	10
検知領域表示	◉ 使用する ○ 使用しない

操作

保存

ストリーミング

表示用URI名	表示する
表示サイズ	480x320 ▼

ストリーミング	
項目	備考
表　示 URI 名	URI プロキシに設定したポート番号が存在する場合、自動入力されます。 また、「表示する」ボタンを押すことによりストリーミング画像が WEB UI 上に表示されます。
表示サイズ	WEB UI 上に表示されるストリーミング画像のサイズを選択できます。

※ WEB UI 上に表示されるストリーミング画像は 5 分間表示されます。

35.4.1 検知イベントトリガー機能

動体検知の動画保存時に、任意のスクリプト内容を実行することができます。スクリプトは、「拡張」→「スクリプト編集」タブから「動体検知イベント」を選択し、編集することができます。

注釈: 動体検知イベントのスクリプトの実行ユーザーは motion となります。そのため、スクリプト内でファイル作成等を行う場合には、パーミッションに注意してください。

尚、本スクリプト向けに環境変数 MOVIE_FILE に検知時に撮影した動画のファイルパスが含まれます。

また、本イベントトリガー用に通知向けの以下のスクリプトを用意しています。これらのスクリプトは、上記の「動体検知イベント」スクリプトに記載し二次的に呼び出して使用します。

通知方法	スクリプトパス	補足
SMS	/var/webui/scripts/motion_send_sms.sh	モデム及び SMS 対応 SIM が必要です。
E-Mail	/var/webui/scripts/motion_send_email.sh	インターネットの経路が必用です。
Unix Domain Socket	/var/webui/scripts/motion_send_socket.sh	

● SMS 通知

OpenBlocks に搭載しているモデム及び SIM を用いて SMS の送信を行います。

以下のファイルが必用となります。

/var/webui/upload_dir/motion_cfg/sendsms.json

サンプル

```
{
    "send_phone_number" : "09012345678" ,
    "send_body" : "動体を検知しました。"
}
```

キー	データ型	内容
send_phone_number	文字列	送信先電話番号
send_body	文字列	送信する SMS 本文

● E-Mail 通知

OpenBlocks が稼働中のネットワークを経由して、メールサーバーに対してメールの送信を行います。以下のファイルが必用となります。

/var/webui/upload_dir/motion_cfg/sendmail.json

サンプル

```
{
    "smtp_server" : "smtp.example.com" ,
    "smtp_port" : 465 ,
    "auth" : true,
    "auth_user" : "exampleuser" ,
    "auth_pass" : "password" ,
    "to" : "from@example.com" ,
    "from" : "from@example.com" ,
    "subject" : "動体検知" ,
    "body" : "動体を検知しました。WEB UI から保存した動画を確認してください。"
}
```

キー	データ型	内容
smtp_server	文字列	SMTP サーバー
smtp_port	INT 型	SMTP ポート
auth	BOOLEAN 型	SMTP-AUTH 使用設定 true : 使用、false : 未使用
auth_user	文字列	認証ユーザー (SMTP-AUTH 使用時に必要)
auth_pass	文字列	認証パスワード (SMTP-AUTH 使用時に必要)
to	文字列	送信先アドレス
from	文字列	送信元アドレス
subject	文字列	件名
body	文字列	メール本文

● Unix Domain Socket 通知

OpenBlocks 内で稼働している Unix Domain Socket 待ち受けを行っているプロセスに対してデータ送信を行うことができます。

以下のファイルが必用となります。

/var/webui/upload_dir/motion_cfg/sendsocket.json

サンプル

```
{
        "socket_path" : "/pd_repeater/userdev_0000001.sock" ,
        "abstract_flag" : true ,
        "send_body" : "動体を検知しました。"
}
```

キー	データ型	内容
socket_path	文字列	Unix Domain Socket パス
abstract_flag	BOOLEAN 型	Abstract Socket 使用設定 true : 使用、false : 未使用
send_body	文字列	送信メッセージ

本機能での送信データのメッセージは JSON となります。

メッセージサンプルは以下となります。

メッセージサンプル

```
{ "time" : "2018-07-17T09:32:15.864+09:00" , "message" : " 動体を検知しました。" }
```

また、以下の PHP ファイルのライブラリを用意することで JSON メッセージデータの拡張が行えます。

/var/webui/upload_dir/motion_cfg/extendsocket.lib.php

拡張 PHP ファイルサンプル
<?php 　　// Default Data Setting from $data 　　$data['extend'] = "追加サンプル" ; ?>

拡張メッセージサンプル
{ "time" : "2018-07-17T09:32:15.864+09:00" , "message" : " 動体を検知しました。" , "extend" : "追加サンプル" }

35.5 メディア機能

「カメラ」→「メディア」タブでは特定ディレクトリ直下に保存した画像及び動画データを表示及びダウンロードすることができます。

表示される画像データ及び動画データの拡張子は以下となります。

種類	対応拡張子
画像データ	.svg、.apng、.bmp、.png、.jpeg、.jpg
動画データ	.mp4、.ogg、.webm

メディアディレクトリパス設定	
項目	備考
メディアディレクトリパス	動体検知等にて保存先に指定したディレクトリを設定します。ディレクトリパス設定後、「リンク」ボタンを押してください。
サムネイル自動生成	メディアディレクトリパスに保存されている画像及び動画に対して、サムネイル画像の生成を行うかを設定します。 設定後、「保存」ボタンを押してください。
画像	画像を表示します。画像をクリックすることでポップアップします。画像のダウンロードに関してはポップアップした画像を各ブラウザの機能からダウンロードを実施してください。
動画	動画を表示します。動画をクリックすることでポップアップします。また、「ダウンロード」ボタンを押すことで対象画像をダウンロードできます。

35.6 ログ機能

「カメラ」→「ログ」タブではストリーミング機能及び動体検知機能のログを確認することができます。

ログ	
項目	備考
ログ	カメラストリーミング機能及び動体検知機能のログファイル一覧が表示されます。確認したいログファイルを選択することで、対象のログファイルが表示されます。 対象のログファイルを選択した場合、ダウンロードボタンが表示されます。ログファイルをダウンロードしたい場合に押してください。

第36章　その他

36.1 カメラ及びストリーミングデータ確認用ユーザー

本機能をインストールした場合、カメラまたは動体検知のストリーミングデータ及び保存した画像・動画データの閲覧のみを許容するユーザーを作成します。このユーザーはカメラ機能のみを閲覧するユーザーであり、OpenBlocks のシステム全体にはアクセスできません。

項目	補足
ユーザー名	"cam_user"となります。
パスワード	デフォルトは"0BSI0T"です。(0 は数字の 0 です。)

※「システム」→「WEB ユーザー」タブにてパスワードの変更が可能です。但し、ユーザー権限の編集を行っても反映されません。

ストリーミング	
項目	備考
表示サイズ	WEB 上に表示するストリーミングデータの画像サイズを選択します。デフォルトでは 0x0 です。 ※ 0x0 の場合、表示されません。
画像	メディア機能で設定したパスの画像データ一覧を表示します。
動画	メディア機能で設定したパスの動画データ一覧を表示します。尚、本機能でのダウンロード機能はありません。

36.2 OpenBlocks IoT VX2 のカメラ接続について

OpenBlocks IoT VX2 に USB カメラを接続する場合、USB type-A ポートに対して直接接続は行わないでください。直接接続を行った場合、USB カメラの動作が不安定となります。そのため、USB ハブ等を経由しての接続を推奨いたします。

36.3 VIDEO デバイスファイルについて

OpenBlocks IoT VX シリーズ及び OpenBlocks IX9 では 1 つの USB カメラを接続した場合、2 つの VIDEO デバイスファイル (ex. /dev/video0 及び/dev/video1) が生成されます。通常では、本機能にて指定する VIDEO デバイスファイルは若番の物 (正しくは、キャプチャ機能対応のデバイスファイル) を設定してください。

尚、OpenBlocks IoT BX 及び EX シリーズでは 1 つの USB カメラを接続した場合、1 つの VIDEO デバイスファイルとなりますので、生成された VIDEO デバイスファイルをご使用ください。

36.4 OpenBlocks IoT BX/EX シリーズへのカメラ設定について

OpenBlocks IoT BX/EX シリーズでは USB カメラを用いてストリーミング機能及び動体検知機能を実施しようとし場合、USB ドライバの問題により以下のようなメッセージを出力し使用できないケースがあります。

この場合、使用時の FPS や解像度を下げることにより使用できますので、FPS 及び解像度を下げての運用をご検討ください。

```
[10554.131243] usb 1-1.4: new high-speed USB device number 5 using dwc3-host
[10554.258997] usb 1-1.4: New USB device found, idVendor=0bad, idProduct=5823
[10554.259029] usb 1-1.4: New USB device strings: Mfr=3, Product=1, SerialNumber=2
[10554.259049] usb 1-1.4: Product: USB Camera
[10554.259067] usb 1-1.4: Manufacturer: Generic
[10554.259085] usb 1-1.4: SerialNumber: 200901010001
[10554.923089] uvcvideo: Found UVC 1.00 device USB Camera (0bad:5823)
[10554.931799] input: USB Camera as /devices/pci0000:00/0000:00:11.0/dwc3-host.2/usb1/1-1/1-1.4/1-
1.4:1.0/input/input2
[10554.932389] usbcore: registered new interface driver uvcvideo
[10554.932408] USB Video Class driver (1.1.1)
[10757.724309] dwc3-host dwc3-host.2: Signal while waiting for configure endpoint command
[10757.741963] usb 1-1.4: Not enough bandwidth for altsetting 0
[10773.412255] dwc3-host dwc3-host.2: xHCI xhci_drop_endpoint called with disabled ep d34ef840
[10773.412288] dwc3-host dwc3-host.2: Trying to add endpoint 0x81 without dropping it.
[10773.412317] usb 1-1.4: Not enough bandwidth for altsetting 7
[10773.412342] dwc3-host dwc3-host.2: xHCI xhci_drop_endpoint called with disabled ep d34ef840
[10804.148391] dwc3-host dwc3-host.2: Trying to add endpoint 0x81 without dropping it.
[10804.148429] usb 1-1.4: Not enough bandwidth for altsetting 7
[11002.887329] dwc3-host dwc3-host.2: Trying to add endpoint 0x81 without dropping it.
[11002.887367] usb 1-1.4: Not enough bandwidth for altsetting 7
```

第X部

Docker設定ガイド

第37章　はじめに

本書は、OpenBlocks シリーズに搭載可能な WBE UI 込みでの Docker(コンテナ型仮想化環境 OSS) の使用方法を解説しています。

第38章　Docker設定

38.1 WEB UI込みDockerのインストール

本製品出荷時では、Docker及びDocker用WEB UIはインストールされておりません。そのため、WEB UIの「メンテナンス」→「機能拡張」タブからDocker及びDocker用WEB UIのインストールを行います。

WEB UIの「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

インストール機能のリストから「Docker(WEB UI 込み)」を選択します。

その後、インストールの「実行」ボタンを押し、インストールを行ってください。

尚、インストール完了後には反映を行うため、本体再起動が必要となります。そのため、「メンテナンス」→「停止・再起動」から本製品の再起動を行ってください。

Docker をインストール際に、一部のドライバのコンパイルを行っております。これにより、インストールに時間がかかりますので、「状況確認」ボタン等にて進行状況をご確認ください。

38.2 WEB UI からの Docker 使用について

Docker(WEB UI 込み) パッケージがインストールされていると、WEB UI の「サービス」→「基本」タブに「Docker」が表示されます。

「サービス」→「基本」タブより「Docker」リンクをクリックすると、ルートタブがサービス用の「ダッシュボード」／「基本」／「Docker」に切り替わります。

※「サービス」→「基本」タブに表示されるリンク及びサービス用のルートタブは、「メンテナンス」→「機能拡張」タブよりインストールされたパッケージにより異なります。

38.3 Docker コンテナ一覧

一度でも起動したことのあるコンテナの状況の確認、起動・停止等の制御を「Docker」→「コンテナ一覧」タブにて行えます。

■コンテナ稼働中の場合

コンテナが稼働中の場合、停止することができます。

停止したいコンテナの「停止」ボタンを押すことにより、停止することが可能です。

尚、稼働中のコンテナは公開ポート及び IP アドレスが表示されます。

■コンテナ停止中の場合

コンテナ停止中の場合、起動及び削除が行えます。

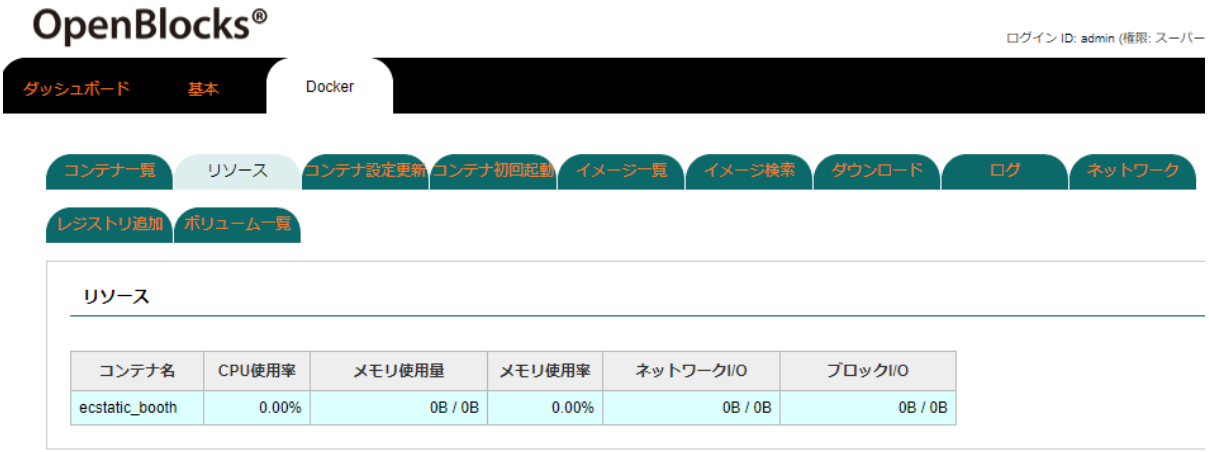
起動させる場合には、起動したいコンテナの「起動」ボタンを押すことで起動します。

対象コンテナを削除する場合には、「削除」ボタンを押してください。

38.4 Docker コンテナリソース表示

一度でも起動したことのあるコンテナのリソース状況確認を「Docker」→「リソース」タブにて行えます。停止中のコンテナは一覧に表示されますが、停止しているため各リソースは未使用状態となります。

また、本項にてリソースを確認し、各コンテナのリソースチューニングを行って下さい。



38.5 Docker コンテナリソース設定更新

コンテナのリソース設定を「Docker」→「コンテナ設定更新」タブにて行えます。

本項では、ホストや他コンテナに影響がでるようなコンテナを使用する場合に、リソースを制限させることを前提としております。

リソース設定を更新するコンテナをプルダウンメニューから選択します。

対象コンテナ選択後、オプションフォーム及び「更新」ボタンを選択できるようになります。

リソース変更用のオプションを入力し、「更新」ボタンを押すことにより適用されます。

本機能では、内部的に docker update コマンドを使用しております。

そのため、オプションフォームには docker update コマンドのリソース制御系のオプション部のみを記載してください。

■最大メモリを 50Mbyte に制限する場合のサンプル	
doceker update コマンド時の例	#docker update --memory 50M --memory-swap 100M nginx_container
オプションフォームに入力する場合の例	--memory 50M --memory-swap 100M

また、docker update コマンドの詳細については以下のページをご確認ください。

http://docs.docker.jp/engine/reference/commandline/update.html

38.6 Docker コンテナ初回起動

コンテナの作成・起動を「Docker」→「コンテナ初回起動」タブにて行えます。

ローカル環境に対象イメージが存在しない場合、自動的に Docker Hub から取得を行います。

※プライベートレジストリサーバのイメージを使用する場合には、「ダウンロード」タブにて事前にイメージを取得してください。

■イメージ

作成及び起動するコンテナのイメージ名を指定します。

■ RUN オプション

docker run コマンドのオプションを指定します。尚、強制的にデタッチオプション (-d) が内部的に付与されます。

Docker 自体はインストール/再起動後、常時起動しております。WEB UI では Docker 本体の再起動等を一部の保存イベントアクションにて行うことがあります。

そのため、使用するコンテナはデタッチ ("docker run -d") 及び常時再起動ポリシー ("docker run --restart=always") オプションを適用し運用してください。

コンテナの管理上、コンテナ名のオプション指定 (--name <コンテナ名>) についても使用を推奨いたします。

また、docker run コマンドの詳細については以下のページをご確認ください。

http://docs.docker.jp/engine/reference/commandline/run.html

38.7 Docker ローカルイメージ確認

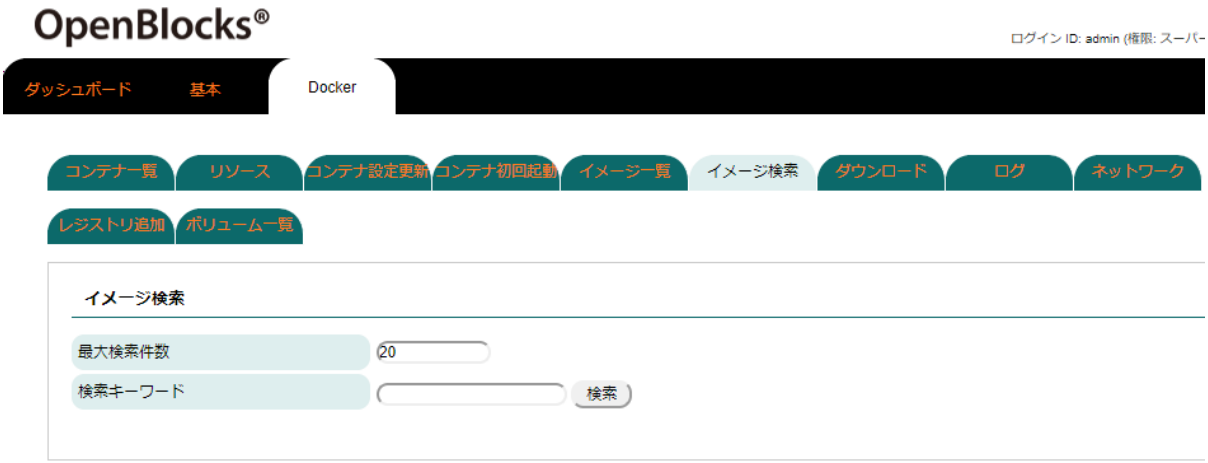
ローカルに存在する Docker コンテナイメージの一覧の確認を「Docker」→「イメージ一覧」タブにて行えます。

コンテナのもととなるイメージの一覧が表示されます。

使用中/未使用ともに「削除」ボタンが表示されますが、使用中のイメージは削除が行えません。そのため、削除を行う場合には、「コンテナ一覧」にて事前に対象イメージを使用しているコンテナを削除してください。

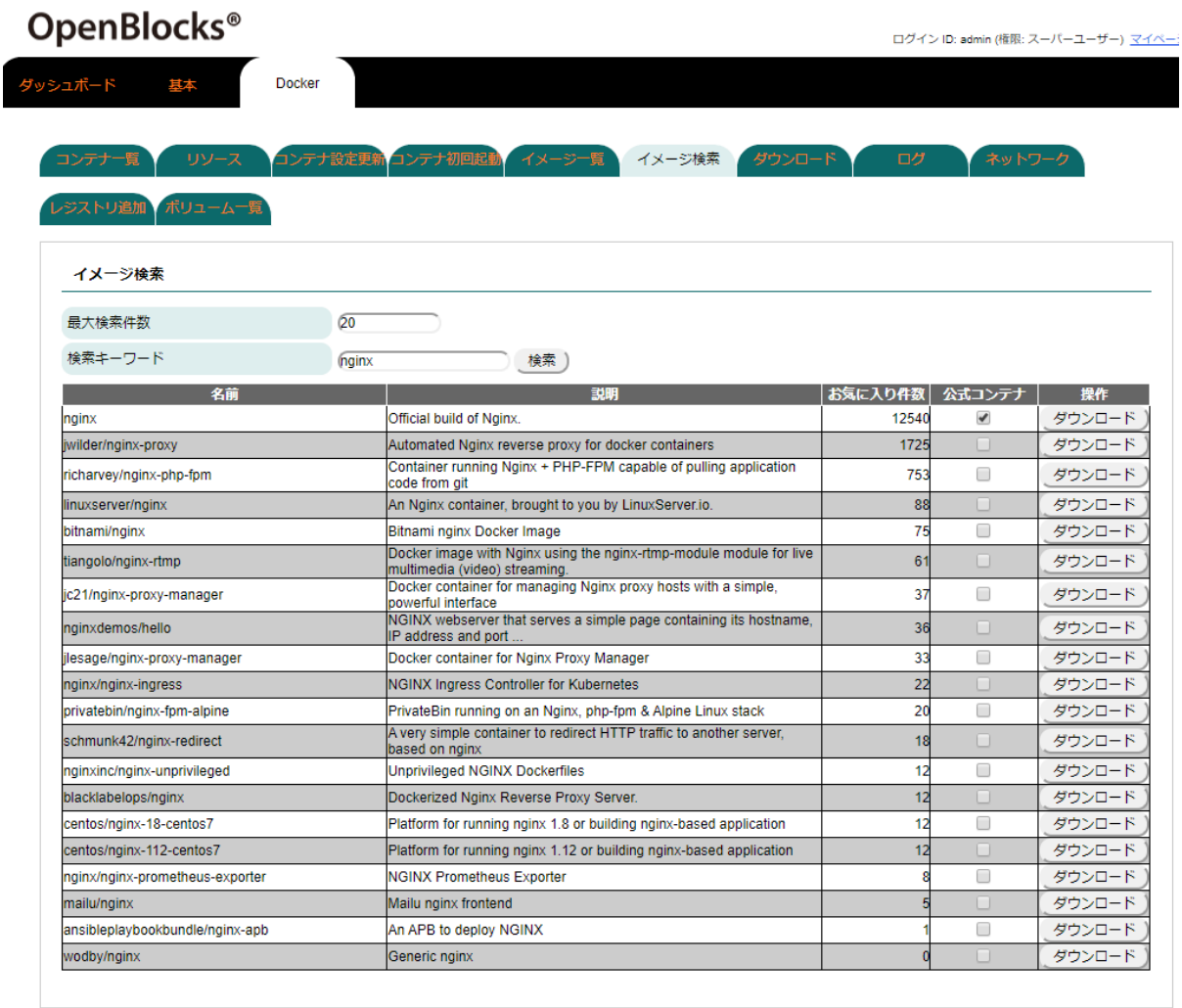
38.8 Docker イメージの検索

Docker Hub に公開されているコンテナイメージを「Docker」→「イメージ検索」タブにて行えます。



検索キーワードフォームに検索対象文字列を入力後、検索ボタンをおすことにより検索されます。

尚、最大検索件数は 100 件までとなっております。



上図は"nginx"のイメージを検索した例となります。

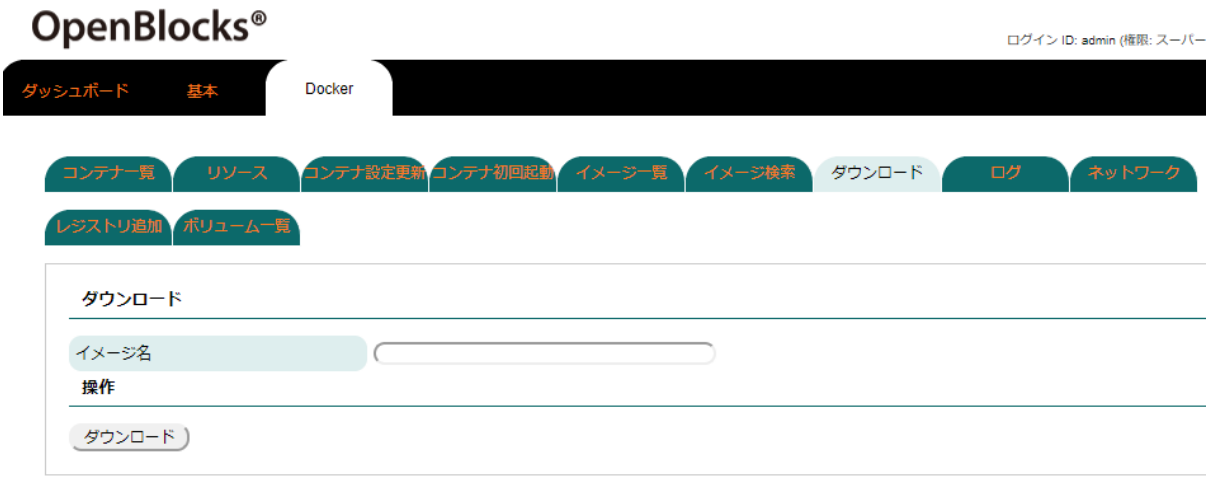
ダウンロードボタンを押すことにより、対象のイメージをダウンロードすることが行えます。

尚、対象イメージをダウンロード中には進捗状況を確認できるよう「検索」ボタンの隣に「状況表示」ボタンが表示されます。こちらのボタンを押して進捗状況の確認を行ってください。

38.9 Docker イメージのダウンロード

Docker Hub に公開されているコンテナイメージやプライベートレジストリサーバに存在するコンテナイメージのダウンロードを「Docker」→「ダウンロード」タブにて行えます。

ダウンロードするコンテナイメージがプライベートレジストリサーバの場合、事前に対象サーバーへのログイン情報が必要となります。こちらについては「レジストリ追加」タブにてログイン情報を設定してください。



ダウンロードしたいイメージ名を指定します。

特定のタグのイメージを取得する場合には、イメージ名 [:タグ名] を指定してください。

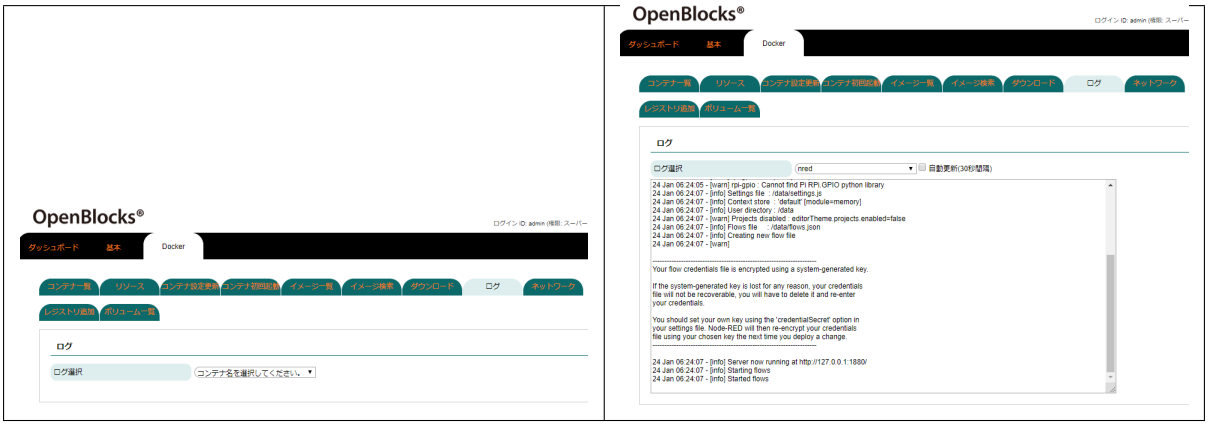
イメージ名入力後、ダウンロードボタンを押すことにより対象のイメージがダウンロードされます。

ダウンロードボタンを押した場合、「状況確認」ボタンが表示されます。このボタンで進行状況が確認できますので、ダウンロードするイメージサイズが大きい場合等にご利用ください。

38.10 Docker コンテナのログ確認

使用しているコンテナが出力しているログの確認を「Docker」→「ログ」タブにて行えます。

尚、コンテナを削除した場合には対象コンテナのログについても削除される為、本項での確認も行えなくなります。



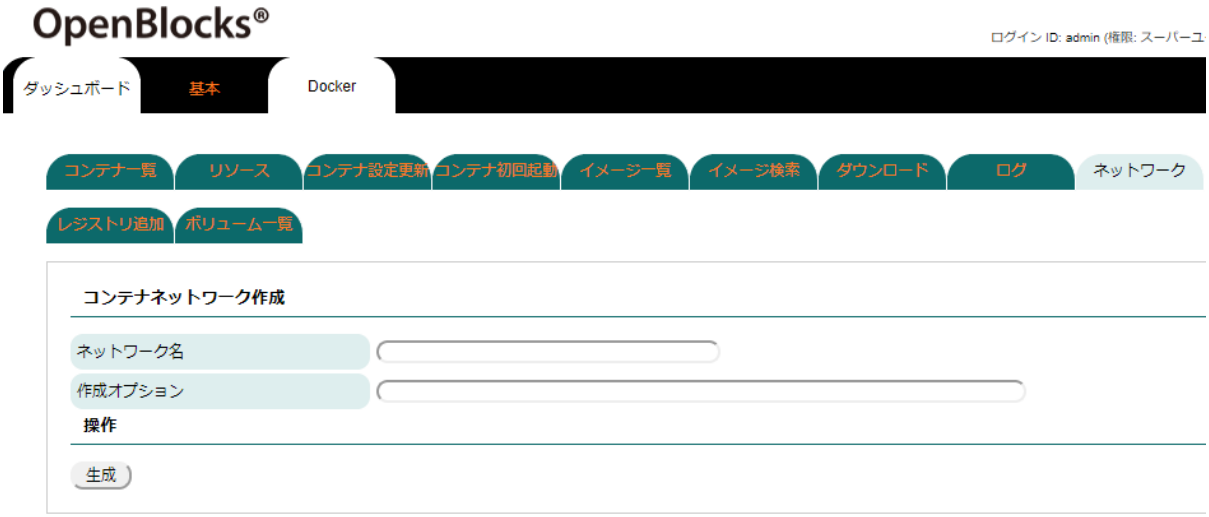
ログを確認したいコンテナをプルダウンメニューから選択します。

コンテナを選択した場合、ログ表示領域が表示されます。

また、対象コンテナの最終ログから一定量表示されます。

38.11 Docker コンテナのネットワーク設定

作成したコンテナをデフォルトのネットワーク以外を用いたい場合、ネットワーク自体の作成及びコンテナとの接続・切断の設定を「Docker」→「ネットワーク」タブにて行えます。



タブを選択すると上図のように表示されます。

■コンテナネットワーク作成 (docker network create コマンド)

作成したいネットワーク名及び作成オプションを指定し、「生成」ボタンを押すことでネットワークが生成されます。

■コンテナネットワーク接続 (docker network connect コマンド)

接続したいネットワーク名及び対象コンテナ、接続オプションを指定し、「接続」ボタンを押すことでネットワークが接続されます。

■コンテナネットワーク切断 (docker network disconnect コマンド)

切断したいネットワーク名及び対象コンテナを指定し、「切断」ボタンを押すことでネットワークが切断されます。

■ネットワーク一覧

Docker のネットワークとして存在しているネットワークの一覧を表示します。作成したネットワークを削除したい場合には、「削除」ボタンを押すことにより削除可能です。

尚、一部のネットワークは Docker の制約上削除することができません。

また、docker netowork 関連のコマンドの詳細は以下のページをご確認ください。

■ docker network create コマンド

http://docs.docker.jp/engine/reference/commandline/network_create.html

■ docker network connect コマンド

http://docs.docker.jp/engine/reference/commandline/network_connect.html

■ docker network disconnect コマンド

http://docs.docker.jp/engine/reference/commandline/network_disconnect.html

38.12 プライベートレジストリサーバーの認証設定

第 3 者へ公開したくないコンテナ等を使用する場合、プライベートレジストリサーバーを用いる場合があります。このプライベートレジストリサーバーを用いる場合、認証情報必要となります。認証情報の設定を「Docker」→「レジストリ追加」タブにて行えます。

ログインしたいプライベートレジストリサーバーに関する情報を設定します。

項目	説明
ログインサーバー	ログインするプライベートレジストリサーバーの FQDN または IP アドレスを指定します。
ユーザー名	ログインする際のユーザー名を指定します。
パスワード	ログイン時のパスワードを指定します。

ログインに成功した場合、上図のようにレジストリ一覧に追加されます。

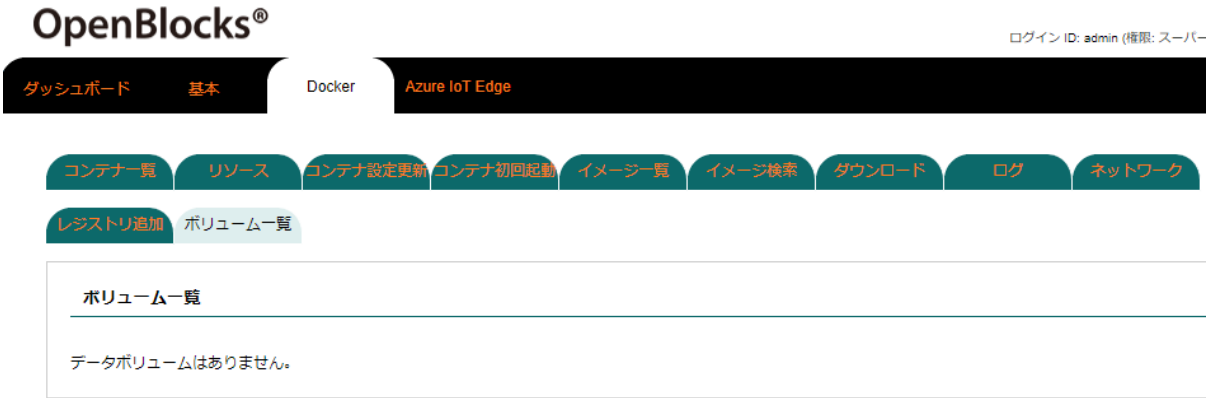
尚、対象レジストリサーバーからログアウトを行う場合には、「ログアウト」ボタンを押して下さい。

ログインボタンを押した段階で実際にログイン処理が適用されます。そのため、プライベートレジストリサーバーへネットワーク接続が行える環境にて実施してください。

38.13 ボリューム一覧表示及び削除

一部のコンテナは、コンテナ及びイメージの削除を行った場合でもストレージ上に存在し続けることがあります。その場合、意図的にボリュームを削除する必要があります。

Docker 管理がディスクボリュームの一覧を「Docker」→「ボリューム一覧」タブにて確認及び削除することができます。



存在している一覧のボリュームを表示します。

「未使用データ削除」ボタンを押すことにより、既に不要となっているボリュームを削除することができます。

※未使用データ削除は Docker DAEMON が不要と判断しているデータを削除します。停止中のコンテナ等についても削除される恐れがありますので、必要なコンテナが稼働中の状態にて実施してください。

第39章　その他

39.1 Azure IoT Edgeとの連携

Azure IoT Edge は Docker を使用しています。Azure IoT Edge 部の WEB UI では、コンテナ自体の直接制御機能はサポートしておりません。Azure IoT Edge の使用を想定している場合には、本機能のご使用についてもご検討ください。

39.2 OpenBlocks IX9での使用について

OpenBlocks IX9 の eMMC ブート時には、使用するコンテナによってストレージが十分なサイズではない可能性があります。そのため、ファイルサイズが大きいコンテナを用いる場合には、SSD のご使用をご検討ください。

第XI部

Azure IoT Edge設定ガイド

第40章　はじめに

本書は、OpenBlocks シリーズに搭載可能な WBE UI 込みでの Azure IoT Edge の使用方法を解説しています。

尚、Azure IoT Edge は、クラウドで行っていた分析とカスタムビジネスロジックをデバイス側で実行できるようにするものです。

第41章　Azure IoT Edge設定

41.1 WEB UI込み Azure IoT Edgeのインストール

本製品出荷時では、Azure IoT Edge 内で使用する Docker、Azure IoT Edge モジュール及び Azure IoT Edge 用 WEB UI はインストールされておりません。そのため、WEB UI の「メンテナンス」→「機能拡張」タブから WEB UI 及び Azure IoT Edge のインストールを行います。

WEB UI の「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

インストール機能のリストから「Azure IoT Edge」を選択します。

その後、インストールの「実行」ボタンを押し、インストールを行ってください。

尚、インストール完了後には反映を行うため、本体再起動が必要となります。そのため、「メンテナンス」→「停止・再起動」から本製品の再起動を行ってください。

Azure IoT Edge のインストールでは、Docker DAEMON が存在しない場合には、DockerDAEMON についてもインストールを行います。インストール作業の内部でコンパイルを行っている為、時間がかかりますので「状況確認」ボタンにてインストール状況をご確認ください。

Azure IoT Edge は内部でコンテナエンジンを使用します。Azure IoT Edge で使用するコンテナの状況等を確認する場合には WEB UI(Docker) 機能の使用についてもご検討ください。

また、ホストマシンとセンサー等のデバイス間でのデータを Azure IoT Edge に対して送信することも可能です。この場合、IoT データ制御機能のご使用についてもご検討ください。

41.2 [事前準備]IoT Hub の設定

Azure IoT Edge を使用する場合、事前に IoT Hub の作成及び IoT Edge デバイスの登録が必要となります。

これらの作業については、以下のページ内の"IoT Hub の作成"、"IoT Edge デバイスを登録する"をご確認の上作業を実施してください。

https://docs.microsoft.com/ja-jp/azure/iot-edge/quickstart-linux

また、登録した IoT Edge デバイスの接続文字列 (プライマリーキーまたはセカンダリーキー) を控えてください。

41.3 WEB UI からの Azure IoT Edge 設定について

Azure IoT Edge パッケージがインストールされていると、WEB UI の「サービス」→「基本」タブに「Azure IoT Edge」が表示されます。

「サービス」→「基本」タブより「Azure IoT Edge」リンクをクリックすると、ルートタブがサービス用の「ダッシュボード」／「基本」／「Azure IoT Edge」に切り替わります。

※「サービス」→「基本」タブに表示されるリンク及びサービス用のルートタブは、「メンテナンス」→「機能拡張」タブよりインストールされたパッケージにより異なります。

41.4 Azure IoT Edge のセットアップ

注釈: 本作業移行以降は"[事前準備]IoT Hub の設定"を実施している必要があります。

WEB UI の「Azure IoT Edge」→「セットアップ」タブを開き、IoT Hub の設定にて控えた接続文字列を使用し、Azure IoT Edge のセットアップを行います。

証明書未使用時	証明書使用時

セットアップ	
「コンフィグ初期化」ボタンにて、設定したコンフィグの初期化が行えます。	
項目	備考
IoT Edge ゲートウェイホスト名	デフォルトで OpenBlocks 本体の FQDN が入力されます。 ※変更する必要はありません。
接続文字列	作成した IoT Edge デバイスの接続文字列を入力します。
証明書設定	証明書を使用するかを設定します。入力として証明書を設定しない場合、内部的に証明書が作成されます。尚、内部生成された証明書を実運用することは推奨されておりません。
デバイス証明書 (証明書使用時)	デバイス CA 証明書のファイルパスを設定します。
デバイスプライベートキー (証明書使用時)	デバイス CA 秘密キーのファイルパスを設定します。
ルート証明書 (証明書使用時)	所有者 CA ファイルパスを設定します。

各証明書ファイルは「システム」→「ファイル管理」タブでのアップロードを推奨いたします。IoT Edge デーモンにて、各証明書のファイル及びファイル格納ディレクトリの権限確認があるため、ファイルには読み取り権限・ディレクトリには実行権限及び読み取り権限を付与する必要があります。

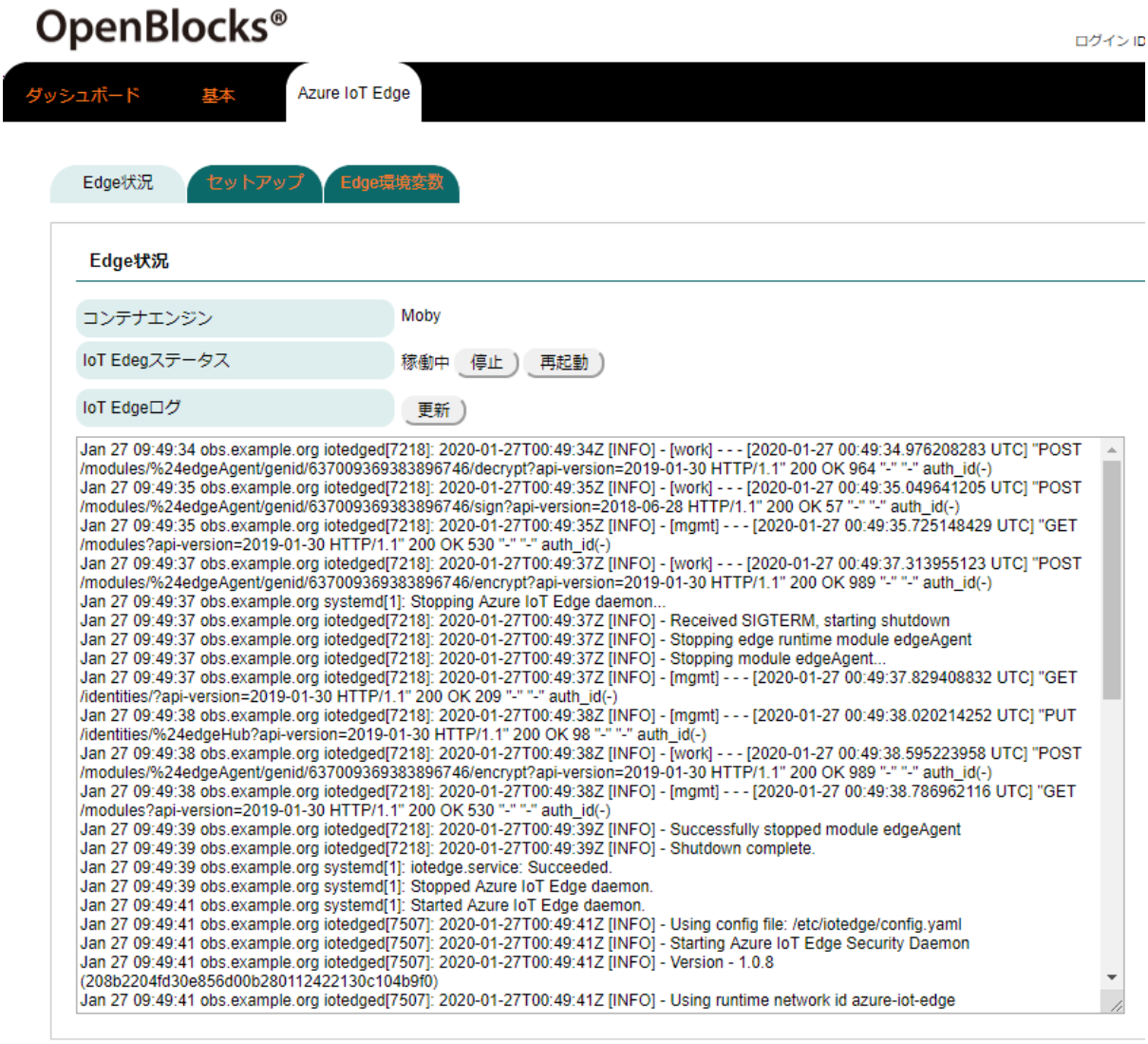
各フォームの入力完了後、「実行」ボタンを押してください。

正常に処理が完了後、「Edge 状況」タブへ遷移します。

41.5 Azure IoT Edge 操作

WEB UI の「Azure IoT Edge」→「Edge 状況」タブでは、以下の作業が行えます。

- コンテナエンジンの確認
- Azure IoT Edge デーモンのステータス状況の確認
- Azure IoT Edge デーモン制御
- Azure IoT Edge 一時ファイルの削除
- Azure IoT Edge のログ確認

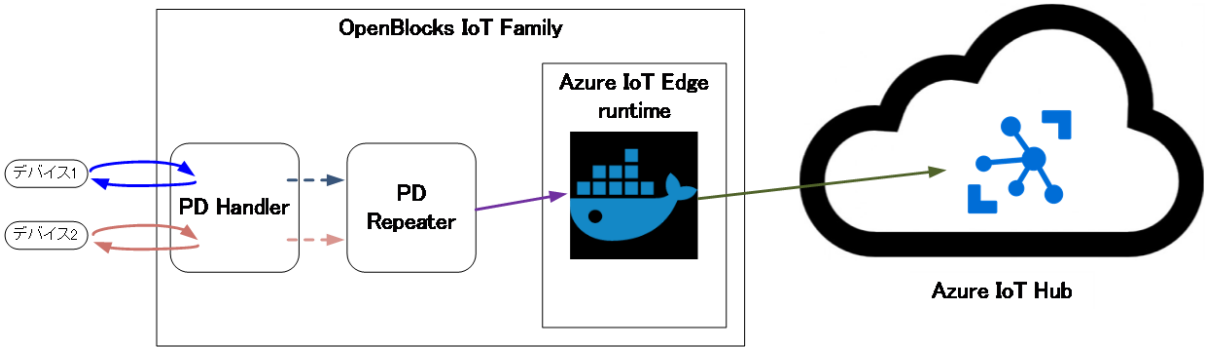

OpenBlocks®
ログイン ID
ダッシュボード
基本
Azure IoT Edge
Edge状況
セットアップ
Edge環境変数
Edge状況
コンテナエンジン
Moby
IoT Edegステータス
稼働中
停止
再起動
IoT Edgeログ
更新
Jan 27 09:49:34 obs.example.org iotedged[7218]: 2020-01-27T00:49:34Z [INFO] - [work] - - - [2020-01-27 00:49:34.976208283 UTC] "POST /modules/%24edgeAgent/genid/637009369383896746/decrypt?api-version=2019-01-30 HTTP/1.1" 200 OK 964 "-" "-" auth_id(-)
Jan 27 09:49:35 obs.example.org iotedged[7218]: 2020-01-27T00:49:35Z [INFO] - [work] - - - [2020-01-27 00:49:35.049641205 UTC] "POST /modules/%24edgeAgent/genid/637009369383896746/sign?api-version=2018-06-28 HTTP/1.1" 200 OK 57 "-" "-" auth_id(-)
Jan 27 09:49:35 obs.example.org iotedged[7218]: 2020-01-27T00:49:35Z [INFO] - [mgmt] - - - [2020-01-27 00:49:35.725148429 UTC] "GET /modules?api-version=2019-01-30 HTTP/1.1" 200 OK 530 "-" "-" auth_id(-)
Jan 27 09:49:37 obs.example.org iotedged[7218]: 2020-01-27T00:49:37Z [INFO] - [work] - - - [2020-01-27 00:49:37.313955123 UTC] "POST /modules/%24edgeAgent/genid/637009369383896746/encrypt?api-version=2019-01-30 HTTP/1.1" 200 OK 989 "-" "-" auth_id(-)
Jan 27 09:49:37 obs.example.org systemd[1]: Stopping Azure IoT Edge daemon...
Jan 27 09:49:37 obs.example.org iotedged[7218]: 2020-01-27T00:49:37Z [INFO] - Received SIGTERM, starting shutdown
Jan 27 09:49:37 obs.example.org iotedged[7218]: 2020-01-27T00:49:37Z [INFO] - Stopping edge runtime module edgeAgent
Jan 27 09:49:37 obs.example.org iotedged[7218]: 2020-01-27T00:49:37Z [INFO] - Stopping module edgeAgent...
Jan 27 09:49:37 obs.example.org iotedged[7218]: 2020-01-27T00:49:37Z [INFO] - [mgmt] - - - [2020-01-27 00:49:37.829408832 UTC] "GET /identities/?api-version=2019-01-30 HTTP/1.1" 200 OK 209 "-" "-" auth_id(-)
Jan 27 09:49:38 obs.example.org iotedged[7218]: 2020-01-27T00:49:38Z [INFO] - [mgmt] - - - [2020-01-27 00:49:38.020214252 UTC] "PUT /identities/%24edgeHub?api-version=2019-01-30 HTTP/1.1" 200 OK 98 "-" "-" auth_id(-)
Jan 27 09:49:38 obs.example.org iotedged[7218]: 2020-01-27T00:49:38Z [INFO] - [work] - - - [2020-01-27 00:49:38.595223958 UTC] "POST /modules/%24edgeAgent/genid/637009369383896746/encrypt?api-version=2019-01-30 HTTP/1.1" 200 OK 989 "-" "-" auth_id(-)
Jan 27 09:49:38 obs.example.org iotedged[7218]: 2020-01-27T00:49:38Z [INFO] - [mgmt] - - - [2020-01-27 00:49:38.786962116 UTC] "GET /modules?api-version=2019-01-30 HTTP/1.1" 200 OK 530 "-" "-" auth_id(-)
Jan 27 09:49:39 obs.example.org iotedged[7218]: 2020-01-27T00:49:39Z [INFO] - Successfully stopped module edgeAgent
Jan 27 09:49:39 obs.example.org iotedged[7218]: 2020-01-27T00:49:39Z [INFO] - Shutdown complete.
Jan 27 09:49:39 obs.example.org systemd[1]: iotedge.service: Succeeded.
Jan 27 09:49:39 obs.example.org systemd[1]: Stopped Azure IoT Edge daemon.
Jan 27 09:49:41 obs.example.org systemd[1]: Started Azure IoT Edge daemon.
Jan 27 09:49:41 obs.example.org iotedged[7507]: 2020-01-27T00:49:41Z [INFO] - Using config file: /etc/iotedge/config.yaml
Jan 27 09:49:41 obs.example.org iotedged[7507]: 2020-01-27T00:49:41Z [INFO] - Starting Azure IoT Edge Security Daemon
Jan 27 09:49:41 obs.example.org iotedged[7507]: 2020-01-27T00:49:41Z [INFO] - Version - 1.0.8 (208b2204fd30e856d00b280112422130c104b9f0)
Jan 27 09:49:41 obs.example.org iotedged[7507]: 2020-01-27T00:49:41Z [INFO] - Using runtime network id azure-iot-edge

Edge 状況	
項目	説明
コンテナエンジン	Azure IoT Edge にて用いるコンテナエンジンを表示します。尚、IoT Edge で用いるコンテナエンジンは Moby を推奨されております。IoT Edge に関してサポートを受ける場合は Moby となっていることを確認してください。
IoT Edge ステータス	IoT Edge デーモンが起動しているか停止しているかを表示します。 また、起動状況に応じて各処理ボタンが表示されます。 ●停止中 起動：IoT Edge デーモンを起動します。 一時ファイル削除：IoT Edge デーモンが内部生成したファイルを削除します。尚、セットアップのコンフィグ情報を変更した場合には、本処理の実施を推奨いたします。 ●稼働中 停止：IoT Edge デーモンを停止します。 再起動：IoT Edge デーモンを再起動します。
IoT Edge ログ	IoT Edge デーモンのログが表示されます。 尚、「更新」ボタンにより最新のログ情報へと更新されます。

セットアップ完了後、本ページから IoT Edge デーモンの起動または本体の再起動によりコンテナのダウンロードが実施されます。その後、Azure IoT Edge が稼働し続けます。

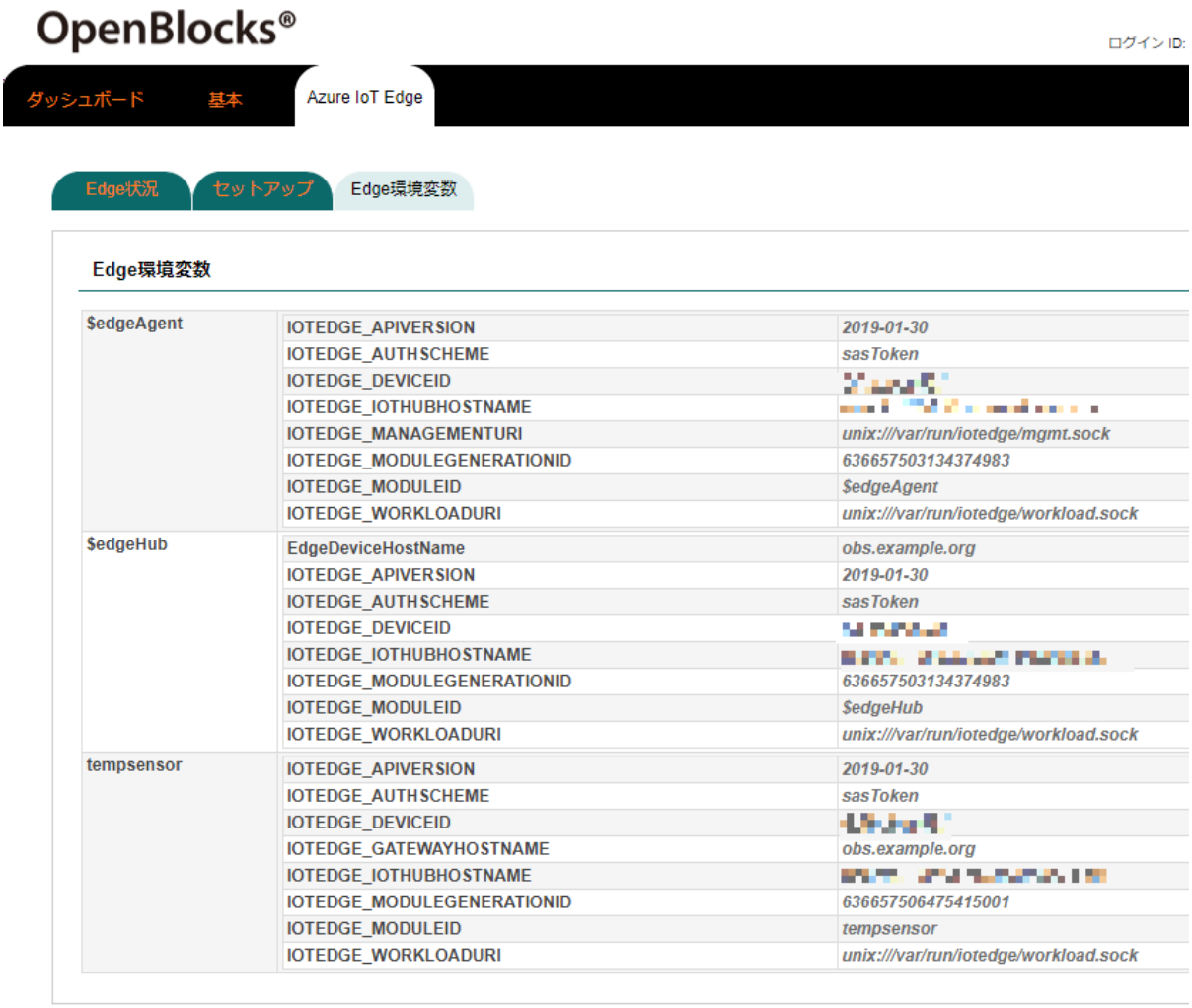
41.6 Azure IoT Edge モジュールの環境変数

下図のように IoT データ制御機能と連携させ、Azure IoT Edge ランタイムにてダウンロードしたコンテナ経由にて Azure IoT Hub へのデータ送信及びフィルタ処理を行うことができます。



IoT データ制御機能にてデバイスデータを IoT Edge モジュールに送信する場合、IoT Edge モジュールが抱えている環境変数を使用する必要があります。

この環境変数は、WEB UI の「Azure IoT Edge」→「Edge 環境変数」タブにて確認を行うことができます。



項目	説明
Edge環境変数	起動後にダウンロードされたコンテナ情報を参照し、Azure IoT Edge 用の環境変数を表示します。尚、IoT データ制御機能にて必要となる情報はモジュール ID(IOTEDGE_MODULEID) と GW ホスト (IOTEDGE_GATEWAYHOSTNAME) です。こちらの情報を IoT データ制御機能の該当タブにて情報を入力してください。

第42章　その他

42.1 WEB UI用Dockerとの連携

本機能ではAzure IoT Edge全体での制御は行えますが、個々のコンテナの制御は行えません。また、追加されたIoT Edgeモジュールや Azure IoT Edge内でのフィルタ機能を用いた場合、使用するメモリ量が過剰な状態となりホストマシン(OpenBlocks)へ影響がでる場合があります。

コンテナ自体のリソース状況や制御機能を行う場合、WEB UI用Dockerをインストールすることによって確認・設定が行えますのでこちらのご使用をご検討ください。

42.2 Azure IoT Edgeのコンテナ追加

Azure FunctionsやAzure ML等の標準以外のコンテナを用意し、デプロイする場合があります。これらのコンテナについては、Azure Container RegistryやDocker Hubから参照しデプロイを行います。以下はAzure Functionsをコンテナ化及びAzure Container Registryを用いたケースの説明です。こちらを参照の上、IoT Edgeのカスタマイズを実施してください。

https://docs.microsoft.com/ja-jp/azure/iot-edge/tutorial-deploy-function#create-a-container-registry

42.3 Azure IoT Edgeのコンテナ追加

Azure FunctionsやAzure ML等の標準以外のコンテナを用意し、デプロイする場合があります。これらのコンテナについては、Azure Container RegistryやDocker Hubから参照しデプロイを行います。以下はAzure Functionsをコンテナ化及びAzure Container Registryを用いたケースの説明です。こちらを参照の上、IoT Edgeのカスタマイズを実施してください。

https://docs.microsoft.com/ja-jp/azure/iot-edge/tutorial-deploy-function#create-a-container-registry

42.4 OpenBlocks IX9での使用について

OpenBlocks IX9のeMMCブート時には、使用するコンテナによってストレージが十分なサイズではない可能性があります。そのため、ファイルサイズが大きいコンテナを用いる場合には、SSDのご使用をご検討ください。

第XII部

FTP設定ガイド

第43章　はじめに

本書は、OpenBlocks シリーズに搭載可能な FTP サーバーソフト及び FTP のクライアントの使用方法を解説しています。

第44章　FTP設定

44.1 FTPのインストール

本製品出荷時では、FTPはインストールされておりません。そのため、WEB UIの「メンテナンス」→「機能拡張」タブからFTPをインストールしてください。

WEB UIの「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張用のパッケージを選択することができます。

インストール機能のリストから「FTP」を選択します。

その後、インストールの「実行」ボタンを押し、インストールを行ってください。

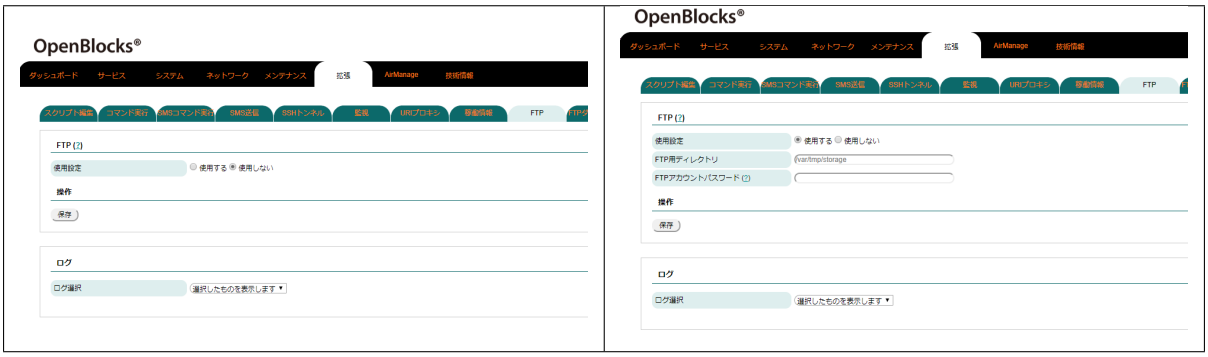
尚、インストール完了後には反映を行うため、本体再起動が必要となります。そのため、「メンテナンス」→「停止・再起動」から本製品の再起動を行ってください。

44.2 FTP サーバー起動設定

FTP のインストールが完了している場合、WEB UI の「拡張」→「FTP」タブに FTP の項目が表示されます。

使用設定を「使用する」に設定し、保存することで FTP サーバーとして扱うことができます。

FTP 用ユーザーとして"obsftpuser"を用意しております。デフォルトパスワードは"0BSI0TF2P"(パスワードの O は数字の 0 です) ですが、WEB UI から設定することでパスワードは変更されます。



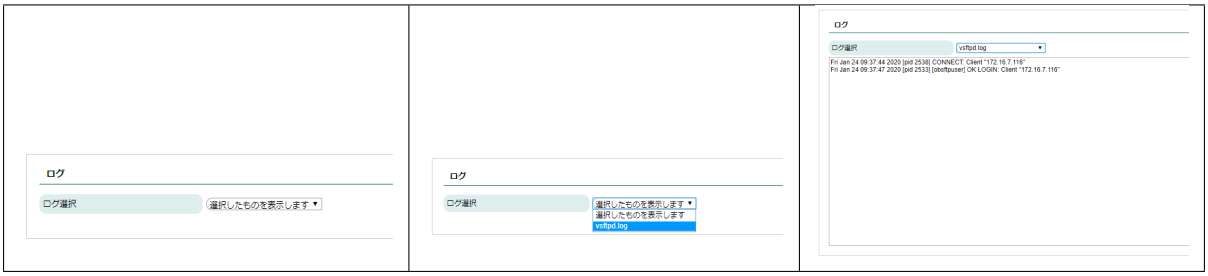
FTP	
項目	説明
使用設定	FTP の使用設定を行います。FTP を使用する場合には、「使用する」を選択してください。
FTP 用ディレクトリ	FTP サーバーログイン時のルートディレクトリのパスを設定します。尚、事前にディレクトリ用意しておく必要がある為、「システム」→「ファイル管理」タブ等にてディレクトリを作成してください。
FTP アカウントパスワード	FTP 用ユーザー"obsftpuser"のパスワードを設定できます。

設定完了後、保存ボタンを押すことにより FTP が起動・停止します。

本機能での書き込み等についてはストレージ寿命を縮めることが想定されます。

そのため、ディレクトリパス先については外部ストレージや tmpfs 等の本体ストレージに影響のない場所を推奨いたします。

また、FTP サーバー機能を使用する状態にて、保存ボタンを押した場合にて、フィルターが解放されていない場合には、解放確認のポップアップが表示されます。また、解放ポリシー設定ポップアップが出ますので、設定したいポリシーのボタンを押してください。



ログ	
項目	説明
ログ選択	ログ選択フォームから該当のログファイルを選択することにより、FTP サーバー機能のログの確認が行えます。 本機能のログは対象ファイルの末尾数十行分のみとなります。 そのため、全内容を確認する場合にはサポートデータを取得しご確認ください。

44.3 FTP パケットフィルタについて

FTP のインストールが完了し FTP サーバー機能を用いる場合、ファイル共有用のプロトコル通信の為 FTP のフィルターを開放する必要があります。FTP サーバー機能を使用しフィルターの開放設定を行っていない場合には、WEB UI の「システム」→「フィルター」タブにてフィルターを開放してください。

デフォルトでは FTP 用のアクセスはできないようにフィルターが適用されています。

"有効"に設定し、保存ボタンを押してください。

44.4 FTP ダウンロード設定

FTP のインストールが完了している場合、WEB UI の「拡張」→「FTP ダウンロード」タブに FTP アップロードの項目が表示されます。

本機能は、対象の FTP サーバーに対して 5 分毎に接続を行いアクセスパスに存在するファイルが前回と同一のものかを確認を行います。ファイルが異なる場合や新規にファイルが置かれている場合には、OpenBlocks 本体へとファイルのダウンロードを行います。

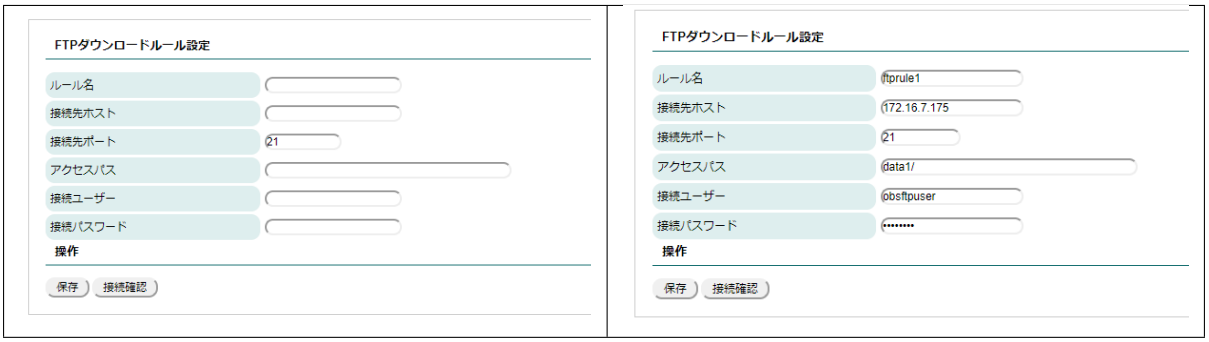
使用設定を「使用する」に設定し保存及び各設定を行うことで自動でファイルが FTP サーバーからダウンロードされます。

※自動ダウンロードを行うため、ダウンロードするファイルのサイズや本体のストレージ容量はチェックしておりません。そのため、大容量のファイルが置かれる場所に対して FTP のアクセスは行わないでください。

※本機能では、FTP プロトコルのサポートのみとなります。FTPS はサポート対象外となります。

FTP ダウンロード	
項目	説明
使用設定	FTP ダウンロードの使用設定を行います。 FTP ダウンロードとして自動でファイルをダウンロードする場合には、「使用する」を選択してください。
保存先ディレクトリ	FTP ダウンロードとして、接続後にファイルをダウンロードした際に保存するディレクトリの指定を行います。 尚、保存されるファイル名は"＜接続先ホスト＞_＜ファイル名＞"また"＜接続先ホスト＞_＜修正アクセスパス＞_＜ファイル名＞"はとなります。 ※保存ファイル名の詳細については、後述部を参照してください。
自動削除設定	FTP にてダウンロードしたファイルの内、一定期間更新等がないファイルを自動的に削除する機能をしようするか設定します。
確保日数	自動削除設定機能にて、変更がないファイルを確保する日数を指定します。

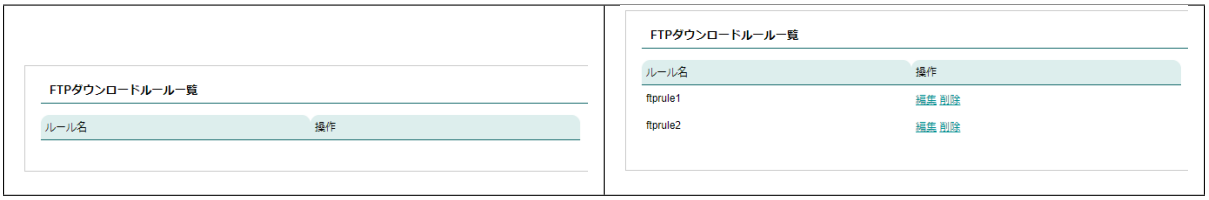
保存ボタンを押すことにより、使用設定及び保存先ディレクトリが適用されます。



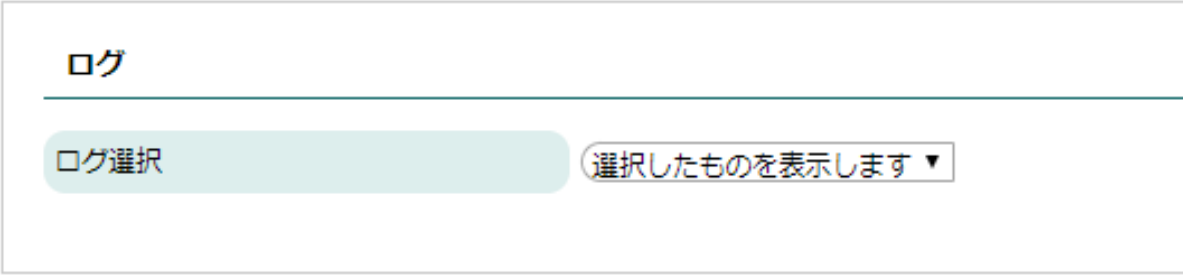
FTP ダウンロードルール設定	
項目	説明
ルール名	FTP ダウンロードする際のユニークな名前を設定します。
接続先ホスト	接続先の FTP サーバーの IP アドレス等の指定を行います。
接続先ポート	FTP プロトコルとして接続するコントロールポートの指定を行います。 ※通常では、デフォルトの 21 から変更する必要はありません。
アクセスパス	接続先ホストに対してログイン後におけるデータをダウンロードしたいディレクトリのパスを設定します。
接続ユーザー	FTP サーバーに接続する際のユーザー名を指定します。
接続パスワード	FTP サーバーに接続する際のユーザー名に対応するパスワードを指定します。

保存ボタンを押すことにより、対象 FTP サーバーへのダウンロード設定が適用されます。

また、接続確認ボタンにより入力中のフォーム値をもとに接続確認が行われます。接続確認結果はボタンの横に表示されます。



FTP ダウンロードルール一覧
FTP ダウンロードルール設定にて保存したダウンロードルールの一覧が表示されます。 「編集」を押すことにより、対象ルールが FTP ダウンロードルール設定のフォームに自動で入力されます。 「削除」を押すことにより、対象ルールが削除されます。



ログ	
項目	説明
ログ選択	ログ選択フォームから該当のログファイルを選択することにより、FTP ダウンロード機能のログの確認が行えます。 本機能のログは対象ファイルの末尾数十行分のみとなります。 そのため、全内容を確認する場合にはサポートデータを取得しご確認ください。

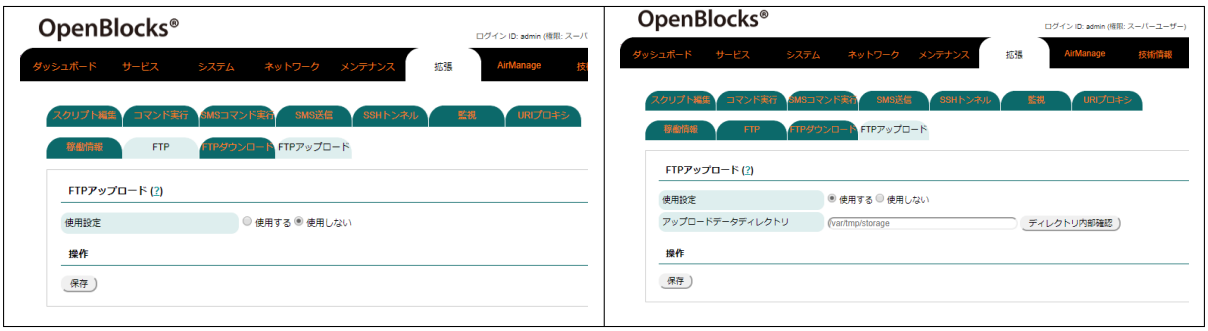
44.5 FTP アップロード設定

FTP のインストールが完了している場合、WEB UI の「拡張」→「FTP ダウンロード」タブに FTP アップロードの項目が表示されます。

本機能は、対象の FTP サーバーに対して 5 分毎に接続を行い対象ディレクトリに存在するルール該当するファイルを FTP サーバーへアップロードを行います。

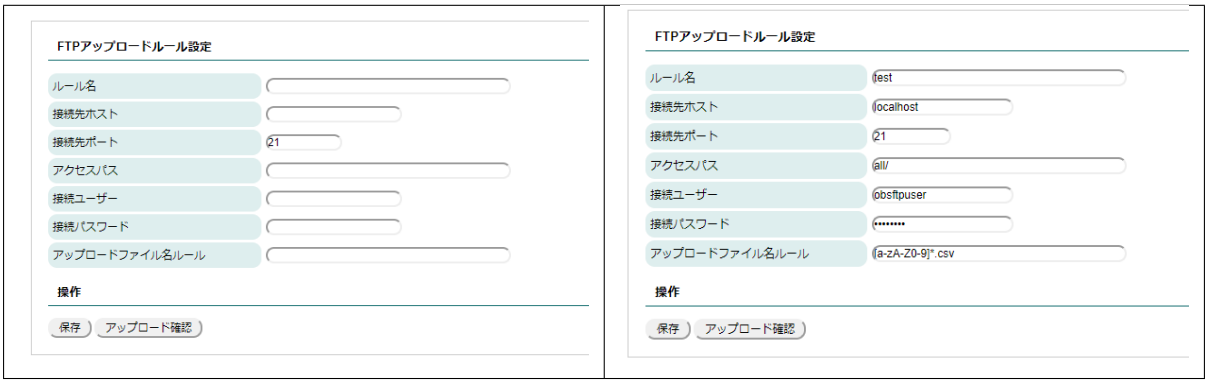
尚、アップロードをおこなったファイルは自動的に削除を行います。

※本機能では、FTP プロトコルのサポートのみとなります。FTPS はサポート対象外となります。



FTP アップロード	
項目	説明
使用設定	FTP アップロードの使用設定を行います。 自動でファイルをアップロードする場合には、「使用する」を選択してください。
アップロードデータディレクトリ	アップロードを行うデータの格納先のディレクトリの指定を行います。 尚、ディレクトリ内部確認ボタンを押すことによりディレクトリ内部のデータの確認が行えます。

保存ボタンを押すことにより、使用設定及びアップロードデータディレクトリが適用されます。

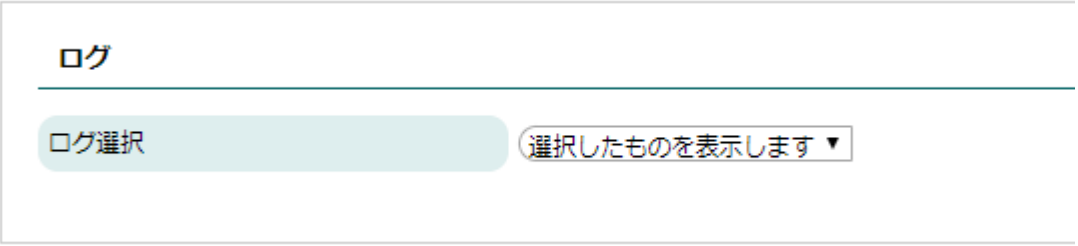


FTP アップロードルール設定	
項目	説明
ルール名	FTP アップロードする際のユニークな名前を設定します。
接続先ホスト	接続先の FTP サーバーの IP アドレス等の指定を行います。
接続先ポート	FTP プロトコルとして接続するコントロールポートの指定を行います。 通常では、デフォルトの 21 から変更する必要はありません。
アクセスパス	接続先ホストに対してログイン後におけるデータのアップロードするディレクトリのパスを設定します。
接続ユーザー	FTP サーバーに接続する際のユーザー名を指定します。
接続パスワード	FTP サーバーに接続する際のユーザー名に対応するパスワードを指定します。
アップロードファイル名ルール	正規表現にアップロード対象のファイル名を指定します。 アップロードファイル名ルールの正規表現は PHP における preg_match 関数を適用しています。 そのため、WEB 等にて preg_match の正規表現の確認を推奨いたします。 尚、入力フォーム部に設定する箇所は preg_match 関数の以下の部分となります。 preg_match('/<入力フォーム部>/' , $filename)

保存ボタンを押すことにより、対象 FTP サーバーへのアップロード設定が適用されます。

また、接続確認ボタンにより入力中のフォーム値をもとに接続確認が行われます。接続確認結果はボタンの横に表示されます。

アップロードルール一覧
FTP アップロードルール設定にて保存したルールの一覧が表示されます。 「編集」を押すことにより、対象ルールが FTP アップロードルール設定のフォームに自動で入力されます。 「削除」を押すことにより、対象ルールが削除されます。



ログ	
項目	説明
ログ選択	ログ選択フォームから該当のログファイルを選択することにより、FTP アップロード機能のログの確認が行えます。 本機能のログは対象ファイルの末尾数十行分のみとなります。 そのため、全内容を確認する場合にはサポートデータを取得しご確認ください。

第45章　その他

45.1 外部ストレージとの連携について

WEB UI の標準機能として、外部ストレージを自動的にマウントする機能が存在しています。ファイルサーバとして扱う場合、一定以上の書き込みが想定されます。そのため、FTP によるファイル共有用の書き込み先については、外部ストレージマウント機能と連携し運用する事を推奨します。

45.2 ストレージクリーナップ機能との連携について

FTP サーバーにおいて書き込み一定以上行い、対象ストレージの使用率が一定以上となるケースが想定されます。WEB UI の標準機能として、ディスクの使用率が一定以上となった場合、特定ディレクトリ配下の一定期間以上前のファイル削除を行うことが可能です。そのため、ストレージクリーンナップ機能と連携し運用する事を推奨します。

45.3 FTP クライアントにおけるファイル名について

FTP クライアント機能にて FTP サーバーからファイルをダウンロードする場合、以下のファイル名規則に基づいてダウンロードされます。

■ダウンロード後のファイル名規則

●アクセスパスを定義している場合

＜接続先ホスト＞_＜修正アクセスパス＞_＜ファイル名＞

●アクセスパスを定義していない場合

＜接続先ホスト＞_＜ファイル名＞

変数	説明
＜接続先ホスト＞	FTP クライアント設定における FTP 設定にて定義した接続先ホスト。
＜修正アクセスパス＞	FTP クライアント設定における FTP 設定にて定義したアクセスパスを修正した値。
＜ファイル名＞	対象 FTP サーバー及びアクセスパス配下に存在するファイルのファイル名。

＜修正アクセスパス＞の修正内容は以下となります。

■修正変換ルール

1, 先頭及び末尾の"/"を削除。

2, 修正後の値から"/"を"_"へと変換。

3, 修正後の値から" "(スペース) を削除。

変数	設定内容	ファイル名
＜接続先ホスト＞	localhost	localhost_aaa_bbb_test.txt
＜修正アクセスパス＞	aaa/bbb/	
＜ファイル名＞	test.txt	
＜接続先ホスト＞	192.168.10.10	192.168.10.10_test_aaa.csv
＜修正アクセスパス＞	test	
＜ファイル名＞	aaa.csv	
＜接続先ホスト＞	192.168.10.10	192.168.10.10_bbb.csv
＜修正アクセスパス＞	(未設定)	
＜ファイル名＞	bbb.csv	

第XIII部

データハンドリング設定リファレンスガイド

第46章　共通事項

46.1 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 等のアプリケーションをモジュールと呼びます。

46.2 設定ファイル

46.2.1 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。

46.2.2 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。

46.3 モジュール間通信

46.3.1 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。

46.3.2 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。

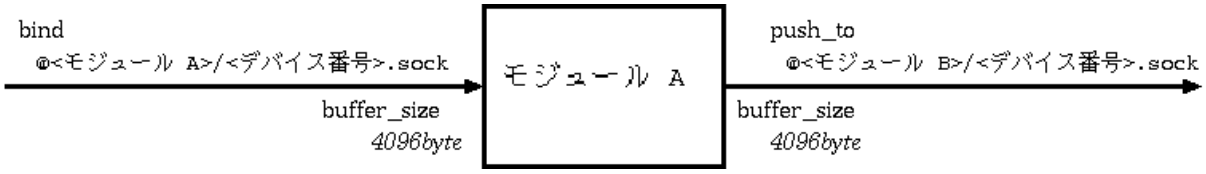
46.3.3 データーサイズ

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。

46.3.4 モジュール間通信に関わるモジュール共通のキー

表 1: 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)



' デバイス番号' は 'localname' と表記される場合もあります。

第47章　PD Repeater

47.1 デォルトパス

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。

表 1: PD Repeater に関連するファイルのデォルトパス

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

47.2 設定ファイルの書式

47.2.1 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": "<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>,
       "truncate": <入力データを受け付けない時間>,
       "receive": <bool 値>,
       "<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>
       }
    ]
}
```

47.2.2 ルートオブジェクト

表 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

47.2.3 servers オブジェクト

表 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

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とも可能です。

PD Excheng(pd_ex)

表 4: PD Exchange (pd ex) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)
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)

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。

汎用 MQTT ブローカー (mqtt)

表 5: 汎用 MQTT ブローカー (mqtt) の設定オブジェクト (1/2)

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. (MAXHOSTNAME-LEN)
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.
forward_match	論理値	受信トピックの前方一致のリファレンスを送られて来たトピックから設定された受信トピックに切り替える. デフォルト値は false.
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. (96byte)
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)
cv_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)

表 6: 汎用 MQTT ブローカー (mqtt) の設定オブジェクト (2/2)

キー	データ型	説明
username	文字列	接続に用いるユーザー名. (96byte)
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)

IBM Watson IoT for device (w4d)

表 7: IBM Watson IoT for device (w4d) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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'messaging.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)

Amazon Kinesis (kinesis)

表 8: Amazon Kinesis (kinesis) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)
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)

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。

PostgreSQL (pgsql)

表 9: PostgreSQL (pgsql) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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 114 (JSONOID).
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)

表 10: MySQL (mysql) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

汎用の Web サーバ (web)

表 11: 汎用の Web サーバ (web) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)
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)

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。

Amazon AWS IoT (awsiot)

表 12: Amazon AWS IoT (awsiot) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

Microsoft Azure Event Hubs (eventhub)

表 13: Microsoft Azure Event Hubs (eventhub) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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'. (MAXHOSTNAMELEN)
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.

IBM Watson IoT for gateway (w4g)

表 14: IBM Watson IoT for gateway (w4g) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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'messaging.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)

Microsoft Azure IoT Hub(iothub)

表 15: Microsoft Azure IoT Hub(iothub) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

NTT docomo Toami for docomo (t4d)

表 16: NTT docomo Toami for docomo (t4d) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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'. (MAXPATHLEN)

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。

UNIX ドメインソケット (lsocket)

表 17: UNIX ドメインソケット (lsocket) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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'. (MAXPATHLEN)

KDDI IoT クラウド Standard(kddi_std)

表 18: KDDI IoT クラウド Standard(kddi_std) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)
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)

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。

NIFTY IoT Device Hub (nf_dvhub)

表 19: NIFTY IoT Device Hub (nf_dvhub) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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 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。

Plat'Home 独自仕様 Web サーバ (pd web)

表 20: Plat'Home 独自仕様 Web サーバ (pd web) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)
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.

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。

Google Cloud IoT Core (iotcore)

表 21: Google Cloud IoT Core (iotcore) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

TCP ソケット (ltcp)

表 22: TCP ソケット (ltcp) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

Amazon AWS IoT [Websocket](awsiot_ws)

表 23: Amazon AWS IoT [Websocket](awsiot_ws) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

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。

Microsoft Azure IoT Hub[Websocket] (iothub_ws)

表 24: Microsoft Azure IoT Hub[Websocket] (iothub_ws) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

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。

ダミー出力 (dummy)

表 25: ダミー出力 (dummy) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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.

SoftBank IoT (sbiot)

表 26: SoftBank IoT (sbiot) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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. (MAXHOSTNAME-LEN)
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 8883.
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 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。

Things クラウド (things)

表 27: Things クラウド (things) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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. (MAXHOSTNAME-LEN)
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_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、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_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)
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。

FTP サーバ (ftp)

表 28: FTP　サーバ (ftp) の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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'.

47.2.4 device オブジェクト

表 29: 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/<localname>.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/<localname>'.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

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。

PD Exchenge(pd_ex)

表 30: PD Exchenge(pd_ex) 固有の設定オブジェクト

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

汎用の MQTT ブローカー (mqtt)

表 31: 汎用の MQTT ブローカー (mqtt) 固有の設定オブジェクト

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

IBM Watson IoT for device (w4d)

表 32: IBM Watson IoT for device (w4d) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

Amazon Kinesis (kinesis)

表 33: Amazon Kinesis (kinesis) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false

PostgreSQL (pgsql)

表 34: PostgreSQL (pgsql) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

表 35: MySQL (mysql) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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'.

汎用の Web サーバ (web)

表 36: 汎用の Web サーバ (web) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false

Amazon AWS IoT (awsiot)

表 37: Amazon AWS IoT (awsiot) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

Microsoft Azure Event Hubs (eventhub)

表 38: Microsoft Azure Event Hubs (eventhub) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

IBM Watson IoT for gateway (w4g)

表 39: IBM Watson IoT for gateway (w4g) 固有の設定オブジェクト

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

Microsoft Azure IoT Hub(iothub)

表 40: Microsoft Azure IoT Hub(iothub) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
device_id	文字列	デバイス ID. (128byte)
device_key	文字列	デバイスキー. (65byte)
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)

NTT docomo Toami for docomo (t4d)

表 41: NTT docomo Toami for docomo (t4d) 固有の設定オブジェクト

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

UNIX ドメインソケット (lsocket)

表 42: UNIX ドメインソケット (lsocket) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false

KDDI IoT クラウド Standard (kddi std)

表 43: KDDI IoT クラウド Standard (kddi std) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false

NIFTY IoT Device Hub (nf_dvhub)

表 44: NIFTY IoT Device Hub (nf_dvhub) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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 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。

Plat'Home 独自仕様 Web サーバ (pd web)

表 45: Plat'Home 独自仕様 Web サーバ (pd web) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

Google Cloud IoT Core (iotcore)

表 46: Google Cloud IoT Core (iotcore) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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)

TCP ソケット (ltcp)

表 47: TCP ソケット (ltcp) 固有の設定オブジェクト

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

Amazon AWS IoT[Websocket](awsiot_ws)

表 48: Amazon AWS IoT[Websocket](awsiot_ws) 固有の設定オブジェクト

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

Microsoft Azure IoT Hub[Websocket](iothub_ws)

表 49: Microsoft Azure IoT Hub[Websocket](iothub_ws) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
device_id	文字列	デバイス ID. (128byte)
device_key	文字列	デバイスキー. (65byte)

ダミー出力 (dummy)

表 50: ダミー出力 (dummy) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false

SoftBank IoT(sbiot)

表 51: SoftBank IoT(sbiot) 固有の設定オブジェクト

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

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。

Things クラウド (things)

表 52: Things クラウド (things) 固有の設定オブジェクト

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

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。

FTP サーバ (ftp)

表 53: FTP サーバ (ftp) 固有の設定オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false
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。

47.3 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。

表 54: メッセージフォーマット

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

表 55: 下流方向メッセージのフォーマット

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 (0x00〜0x03)
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

Cloud ID とヘッダーの内容を示します。

表 56: Cloud ID とヘッダーの内容

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 *1
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 *1
MS Azure IoT Hub	0x0a	対応	MQTT 受信トピック *2
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 *3
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 *2
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 *4
FTP サーバ	0x17		

1. 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。

2. 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。

3. 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> }

4. 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。

47.4 Plat ’Home 独自仕様 WEB サーバ

47.4.1 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	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

47.4.2 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。

第48章　PD Broker

48.1 デォルトパス

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。

表 1: PD Broker に関連するファイルのデォルトパス

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

48.2 設定ファイルの書式

48.2.1 構文

```
{
    "brokers": [
       {
           "enable" : <bool 値>,
           "buffer_size" : <バッファサイズ>,
           "topic_size" : <トピックサイズ>,
           "bind": "<受信ソケット名>",
           "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>,
           "buffer_size" : <バッファサイズ>,
           "topic_size" : <トピックサイズ>,
           "bind": "<受信ソケット名>",
```

(次のページに続く)

(前のページからの続き)

```
            "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>"
                }
            ]
        }
    ]
}
```

48.2.2 broker オブジェクト

表 2: broker 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。'@/<module name>/<localname>.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

48.2.3 destinations オブジェクト

表 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.

第49章　PD Agent

49.1 デォルトパス

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。

表 1: PD agent に関連するファイルのデォルトパス

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

49.2 設定ファイルの書式

49.2.1 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 ”:[
                {
```

(次のページに続く)

(前のページからの続き)

```
                    <channels オブジェクト>
                }
            ]
        }
    ]
}
```

49.2.2 agent オブジェクト

表 2: agent 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	制御メッセージ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.
channels	JSON 配列	channels オブジェクト. 最大 32 個まで設定可能.

49.2.3 channels オブジェクト

表 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
index	文字列	'exec' キーに指定される実行オブジ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ジ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ジェクトもしくは Dynamic Link モジュール又は Lua 言語スクリプトのパス名のパス名. ファイル記述子が .so の場合は Dinamic Link モジュール、.lua の場合は Lua 言語スクリプト、それ以外の場合は実行オブジェクトと見なされる.(MAXPATHLEN)
args	文字列	'exec' キーに設定される実行オブジェクトに与える引数. 'exec' に指定されるパス名が Dynamic Link モジュールまたは Lua 言語スクリプト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.

49.2.4 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。

表 4: 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

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)

49.2.5 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。　SONE オブジェクトポインタ 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。

```
/*
 *  Copyright (c) 2018 - 2020
 *       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 <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>
#ifdef __linux__
#include <lua5.3/lua.h>
#else
#include <lua.h>
#endif
```

(次のページに続く)

(前のページからの続き)

```
#define N_CHANNEL        32

struct channel_t {
        unsigned char        reply;                        /* Enable replay or not.
↪                        */
        unsigned char        objtype;                /* 1 for dynamic link module.
↪                        */
        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.                */
        char                 lua_func[64];                /* Not use in 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.                 */
        lua_State        *lua_state;                /* Not use in dynamic link␣
↪module.                */
};

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.
```

(次のページに続く)

(前のページからの続き)

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

/*
        Your own exec function : int dl_exec()
            JSON *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(json_t *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.
```

(次のページに続く)

(前のページからの続き)

```
//        Set any value for 'result' key

        json_object_set(result, "done", json_true());
        json_object_set(result, "localname", json_string(agent->localname));

        return(0);
}
```

49.2.6 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 SONE 文字列を返して下さい。

```
--
--  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.
```

(次のページに続く)

(前のページからの続き)

```
-- 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.
```

(次のページに続く)

(前のページからの続き)

```
        result = string.format("{\"done\": true, \"localname\":\"%s\"}", localname)

        return 0, result
end
```

49.2.7 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。

実行コマンドを起動した

```
{
   "time": "timestamp", "reply_to": "md5", "result": "queuing",
   "reason": "matched", "matched": {"key": "value"}
}
```

実行コマンドの実行に失敗した

```
{
   "time": "timestamp", "reply_to": "md5", "result": "faild",
   "reason": "matched", "matched": {"key": "value"}
}
```

実行コマンドの実行を終了した

```
{
   "time": "timestamp", "reply_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", "reply_to": "md5", "result": "not queuing",
   "reason": "key and value not matched"
}
```

制御メッセージが JSON 文字列でない

```
{
   "time": "timestamp", "reply_to": "md5", "result": "not queuing",
   "reason": "not JSON form"
}
```

ここで、md5 は、*PD Repeater* **の下流方向メッセージ** に記載される制御メッセージのハッシュ値、{"key": "value"} は、一致した「判断キー」と「処理判断値」です。

実行ステータスは、実行オブジェクトもしくはシェルスクリプトを呼び出す execv() の戻り値であり、実行オブジェクトもしくはシェルスクリプトの処理結果を示すものではありません。　処理の完全性を求めるのであれば応答メッセージは実行オブジェクトもしくはシェルスクリプトから agent_push_to 環境変数で継承される UNIX ドメインソケットに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) に置き換えられる.

Lua 言語スクリプト使用時で、戻り値 Stack2 が NULL で無い場合、実行終了時の 'result' キーの値は、戻り値 Stack2 の値 (JSON 文字列) に置き換えられる.

第50章　PD Handler BLE (Node.js)

50.1 デォルトパス

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。

表 1: PD Handler BLE (Node.js) に関連するファイルのデォルトパス

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.)

50.2 設定ファイルの書式

50.2.1 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>
}
```

50.2.2 ルートオブジェクト

表 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

50.2.3 servers オブジェクト

表 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

50.2.4 beacon オブジェクト

表 4: beacon 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
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/<localbame>.sock' が設定されます.
duplicate_type	文字列	'interval', 'entry', 'inout' のいずれか.
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	payload_manage オブジェクト
data_filter_rule	JSON 配列	data_filter_rule オブジェクト
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

50.2.5 payload_manage オブジェクト

表 5: 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

50.2.6 data_filter_rule オブジェクト

表 6: 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

50.2.7 blesensor オブジェクト

表 7: blesensor 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
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/<localbame>.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

第51章　PD Handler BLE (C)

51.1 デォルトパス

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。

表 1: PD Handler BLE (C) に関連するファイルのデォルトパス

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

設定ファイルは *PD Handler BLE (Node.js)* と同じです。
詳細は *PD Handler BLE (Node.js)* の **設定ファイルの書式** を参照してください。

第52章　PD Handler UART

52.1 デォルトパス

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。

表 1: PD Handler UART に関連するファイルのデォルトパス

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

52.2 設定ファイルの書式

52.2.1 EnOcean

構文

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

ルートオブジェクト

表 2: ルートオブジェクト

キー	データ型	説明
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

enocean オブジェクト

表 3: 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_device オ ブジェクト

表 4: enocean_device 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
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/<localbame>.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

52.2.2 Wi-SUN B ルート

構文

```
{
    "prototype": "wisun",
    "wisun": <wisun オブジェクト>
}
```

ルートオブジェクト

表 5: ルートオブジェクト

キー	データ型	説明
キー	データ型	説明
prototype	文字列	'wisun'
wisun	JSON obj	wisun オブジェクト

wisun オブジェクト

表 6: wisun 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
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/<localbame>.sock' が設定されます.
serial_port	文字列	デフォルトは "/dev/ttyEX2"
routeb_id	文字列	B ルート認証 ID、32 桁の 16 進数（0〜9, A〜F).
password	文字列	B ルート認証のパスワード、英数字 12 文字（大文字・小文字区別なし）

第53章　PD Handler HVSMC

53.1 デォルトパス

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。

表 1: PD Handler HVSMC に関連するファイルのデォルトパス

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

53.2 設定ファイルの書式

53.2.1 構文

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

53.2.2 ルートオブジェクト

表 2: ルートオブジェクト

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

53.2.3 hvsmc オブジェクト

表 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
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/<localbame>.sock' が設定されます.
memo	文字列	出力に付加されるユーザー定義文字列. (256byte)
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

第54章　PD Handler Modbus

54.1 Modbus ファンクションコード

表 1: PD Handler Modbus に用いられている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 ビット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	クライアント／サーバー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 ビット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（サーバー）機器の ID を通知する。
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	Mark 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。

0x09, 0x0a, 0x13, 0x14 物理的な入力を持たない 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 プロトコルには存在しない独自拡張のファンクションです。

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。

表 2: “ function ”キーに用いるファンクションコードの別称

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_input_register
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_register
0x11	Report Slave ID	17	report_slave_id
0x13	Write Multiple Discrete Input	19	write_muliple_discrete_input
0x14	Write Multiple Input Registers	20	write_multiple_input_registers
0x16	Mark Write Registers	22	mark_write_registers
0x17	Write and Read Registers	23	write_and_read_registers

54.2 PD Handler Modbus Client

54.2.1 デォルトパス

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。

表 3: PD Handler Modbus Client に関連するファイルのデォルトパス

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

54.2.2 設定ファイルの書式

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>
                }
        },
    ]
}
```

ルートオブジェクト

表 4: 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

clients オブジェクト

clients オブジェクトは、PD Handler Modbus Client の動作を規定する設定オブジェクトです。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。

表 5: clients 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. (32byte)
bind	文字列	制御メッセージを受け取るソケット名. 文字列の先頭が '@' の場合は abstract namespace と解釈する. 空の場合は、デフォルト値 '@/pd_handler/<localname>.sock' が設定される. (MAXPATHLEN)
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 と解釈する. 空の場合は、デフォルト値 '@/pd_repeater/<localname>.sock' が設定される. (MAXPATHLEN)
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
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.
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'.
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_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'
node	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	文字列	【RTU 接続】のデバイス名. デフォルト値は '/dev/tty00'
rtu_speed	整数値	【RTU 接続】のビットレート. デフォルト値は 115200
rtu_bits	整数値	【RTU 接続】のビット数. 8 又は 7. デフォルト値は 8.
rtu_parity	文字列	【RTU 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	【RTU 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	【RTU 接続】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	文字列	出力に付加されるユーザー定義文字列. (256byte)
infos	JSON Object	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

acquisitions オブジェクト

表 6: 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 接続のみ)
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.

54.2.3 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。

表 7: PD Handler Modbus Client で利用可能なファンクションコード

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		○

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。

54.2.4 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
```

表 8: PD Handler Modbus Client の CSV ファイルの書式

パラメータ	データの形式	説明
デバイス番号	半角英数字	WebUI により割り振れたデバイス番号を記載します。WebUI に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。

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。

54.2.5 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。

表 9: 基準時刻制御における取得時間間隔の設定と実動作値

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」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。

54.2.6 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。

表 10: PD Handler Modbus Client のメッセージオブジェクト

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 接続時のみ
port	ポート番号	整数値	入出力	TCP 接続時のみ.
device	デバイスファイル名	文字列	入出力	RTU 接続時のみ
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	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	

54.2.7 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]
}
```

54.3 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。

表 11: PD Handler Modbus Server のレジスタマップ

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

54.3.1 デォルトパス

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。

表 12: PD Handler Modbus Server に関連するファイルのデォルトパス

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）
/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

54.3.2 設定ファイルの書式

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>
        }
    ]
}
```

ルートオブジェクト

表 13: 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

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。

表 14: servers 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. (32byte)
bind	文字列	制御メッセージを受け取るソケット名. 文字列の先頭が '@' の場合は abstract namespace と解釈する. 空の場合は、デフォルト値 '@/pd_handler/<localname>.sock' が設定される. (MAXPATHLEN)
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 と解釈する. 空の場合は、デフォルト値 '@/pd_repeater/<localname>.sock' が設定される. (MAXPATHLEN)
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.
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.
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.
node	文字列	【TCP 接続】PLC 機器からの接続を待ち受ける IP アドレス. デフォルト値は'0.0.0.0'
port	整数値	【TCP 接続】PLC 機器からの接続を待ち受ける TCP ポート番号. デフォルト値は 502
device	文字列	【RTU 接続】PLC 機器からの接続を待ち受けるシリアル接続デバイス名. デフォルト値は '/dev/tty00'
rtu_speed	整数値	【RTU 接続】PLC 機器からの接続を待ち受けるシリアル接続ビットレート. デフォルト値は 115200
rtu_bits	整数値	【RTU 接続】PLC 機器からの接続を待ち受けるシリアル接続ビット数. 8 又は 7. デフォルト値は 8.
rtu_parity	文字列	【RTU 接続】PLC 機器からの接続を待ち受けるシリアル接続パリティ. 'none','even','odd' のいずれか. デフォルト値は'none'.
rtu_stop	整数値	【RTU 接続】PLC 機器からの接続を待ち受けるシリアル接続ストップビット. 1 又は 2. デフォルト値は 1.
rtu_rts_delay	整数値	【RTU 接続】PLC 機器からの接続を待ち受けるシリアル接続の 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	文字列	出力に付加されるユーザー定義文字列. (256byte)
infos	JSON Object	出力に付加されるユーザー定義オブジェクト

54.3.3 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。

表 15: PD Handler Modbus Server で利用可能なファンクションコード

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	○	○

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。

54.3.4 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]
}
```

54.3.5 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
```

第55章　PD Handler Modbus 2

55.1 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。

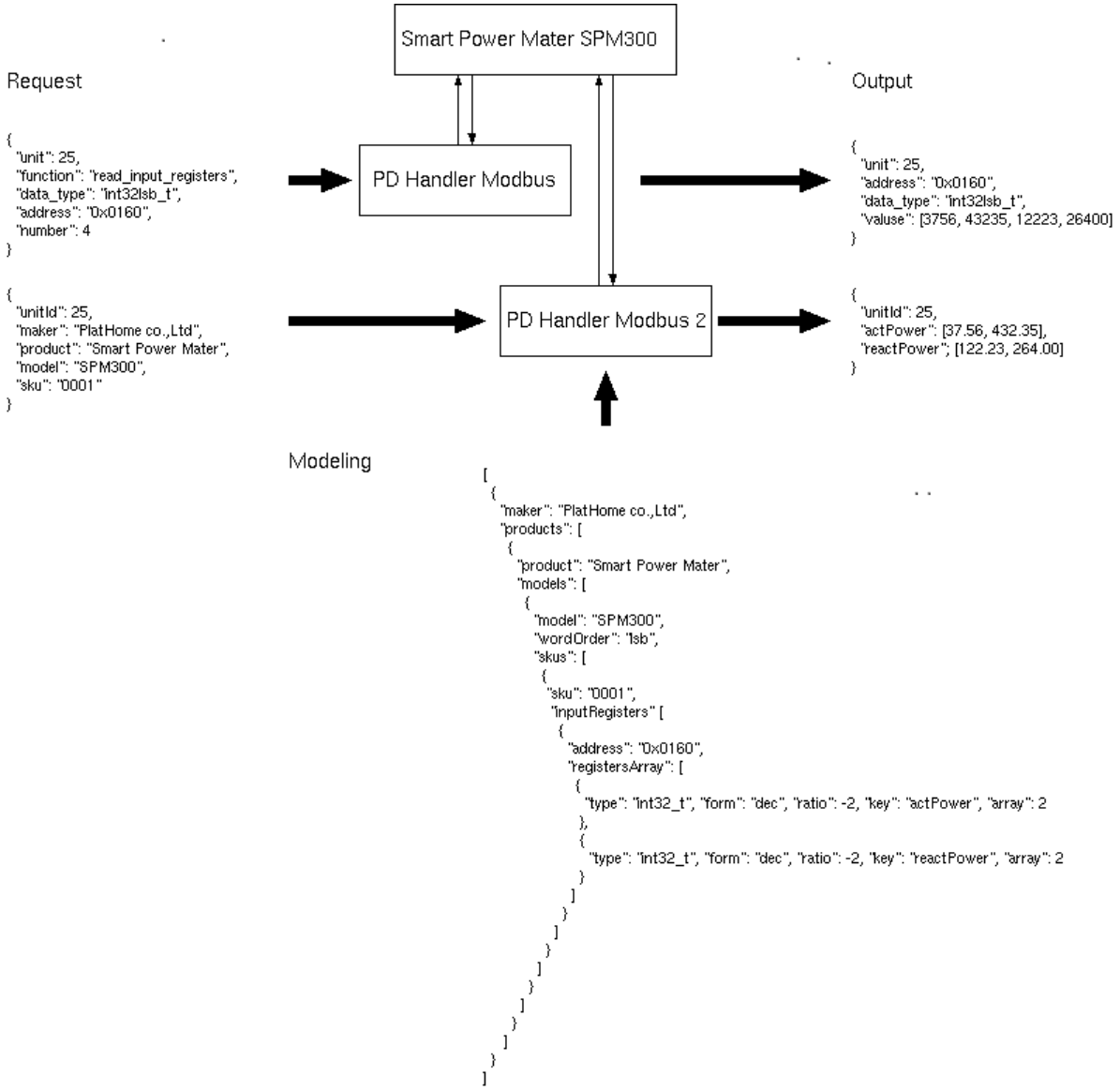


図 1: PD Handler Modbus と PD Handler Modbus 2 の比較

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。

55.1.1 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。

55.1.2 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,
```

(次のページに続く)

(前のページからの続き)

```
    "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,
```

(次のページに続く)

(前のページからの続き)

```
                                    "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。

55.1.3 Modbus モデリングファイルの書式

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,
                                              :
```

(次のページに続く)

(前のページからの続き)

```
                                    }
                                ],
                                "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,
            :
    }
]
```

ルートオブジェクト

表 1: 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)

maker オブジェクト

表 2: 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)

product オブジェクト

表 3: 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)

model オブジェクト

表 4: 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 (μsec). 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)

sku オブジェクト

表 5: sku オブジェクト

キー	データ型	説明
sku	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
rawRequest	JSON 配列	RAW リクエストのための情報. rawRequest 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.

bit オブジェクト

表 6: 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)

register オブジェクト

表 7: 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)

rawRequest オブジェクト

表 8: 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'

bitsArray オブジェクト

表 9: bitsArray オブジェクト

キー	データ型	説明
include	論理値	こ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.

registersArray オブジェクト

表 10: 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'.
base_ratio	整数値	値の係数に用いる冪乗値の底を指定する. デフォルト値は 10.
ratio	整数値	値の係数に用いる冪乗値の冪指数を指定する. 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.

55.1.4 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。

表 11: モデリングファイル作成支援ツール

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

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.
```

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。

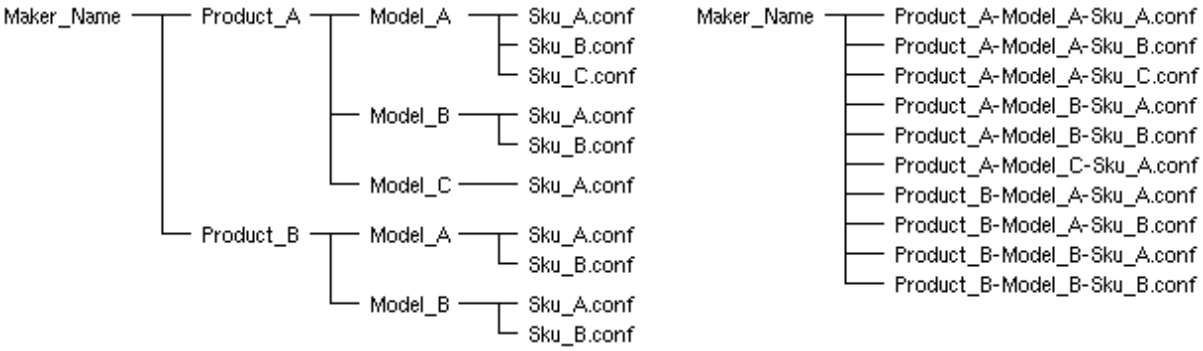


図 2: ディレクトリ階層を細分化した例とフラット展開した例

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.
```

(次のページに続く)

(前のページからの続き)

```
    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.
```

55.2 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。

55.2.1 デォルトパス

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

表 12: PD Handler Modbus 2 Clien に関連するファイルのデォルトパス

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_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

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。

55.2.2 設定ファイルの書式

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。

```
{
    "csv_file": "<CSV ファイル>",
    "model_file": "<モデリングファイル>",
    "clients" : [
        {
            <clients オブジェクト>
        },
        {
            <clients オブジェクト>
        }
    ]
}
```

ルートオブジェクト

表 13: 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_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)

clients オブジェクト

clients オブジェクトは、PD Handler Modbus 2 Client の動作を規定する設定オブジェクトです。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。

表 14: clients 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. (32byte)
bind	文字列	制御メッセージを受け取るソケット名. 文字列の先頭が '@' の場合は abstract namespace と解釈する. 空の場合は、デフォルト値 '@/pd_handler/<localname>.sock' が設定される. (MAXPATHLEN)
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 と解釈する. 空の場合は、デフォルト値 '@/pd_repeater/<localname>.sock' が設定される. (MAXPATHLEN)
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を行うか否か. 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.
node	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	文字列	【RTU 接続】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	【RTU 接続】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	【RTU 接続】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	【RTU 接続】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	【RTU 接続】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	【RTU 接続】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.
rtu_reset	整数値	【RTU 接続】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.
id_form	文字列	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 Object	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
acquisitions	JSON 配列	acquisitions オブジェクト (最大 32 オブジェクト)

acquisitions オブジェクト

表 15: 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.

55.2.3 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。

CSV ファイルの書式は、次の通りです。

```
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。

記載例

```
#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
mdc2dev_0000001,2,1,"Toukou Toushiba","SmaMe Series",S2MS-RNS22,0001,IBS,1-1,Mater,
↪2,300,true,00:00
mdc2dev_0000001,3,1,"Toukou Toushiba","SmaMe Series",S2MS-RNS22,0001,IBS,1-2,Mater,
↪1,300,true,00:00
mdc2dev_0000001,4,1,"Toukou Toushiba","SmaMe Series",S2MS-RNS22,0001,IBS,1-2,Mater,
↪2,300,true,00:00
```

55.2.4 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。

表 16: 基準時刻制御における取得時間間隔の設定と実動作値

取得時間間隔の設定値	実動作値
0 〜 199	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」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。

55.2.5 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。

表 17: PD Handler Modbus 2 Client のメッセージオブジェクト

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. キーの名称は 'timestamp_key' で変更可能.
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.

55.2.6 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', 'array' に応じた値でなくてはなりません。但し書き込み処理において ’base_ratio ’と ’ratio ’を用いた数値の逆変換は行われません。

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。

55.2.7 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', 'array' に応じた値でなくてはなりません。但し書き込み処理において ’base_ratio ’と ’ratio ’を用いた数値の逆変換は行われません。

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
}
```

55.3 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。

55.3.1 デォルトパス

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。

表 18: PD Handler Modbus 2 Server に関連するファイルのデォルトパス

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_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/<localname>.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

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。

55.3.2 設定ファイルの書式

構文

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

```
{
    "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>
        }
    ]
}
```

ルートオブジェクト

表 19: 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)
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)

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。

表 20: serverss 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. (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/<localname>.map
bind	文字列	制御メッセージを受け取るソケット名. 文字列の先頭が '@' の場合は abstract namespace と解釈する. 空の場合は、デフォルト値 '@/pd_handler/<localname>.sock' が設定される. (MAXPATHLEN)
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 と解釈する. 空の場合は、デフォルト値 '@/pd_repeater/<localname>.sock' が設定される. (MAXPATHLEN)
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を行うか否か. 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.
node	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	文字列	【RTU 接続】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	【RTU 接続】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	【RTU 接続】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	【RTU 接続】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	【RTU 接続】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	【RTU 接続】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.
rtu_reset	整数値	【RTU 接続】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.
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.
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.
id_form	文字列	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 Object	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

55.3.3 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。

表 21: PD Handler Modbus 2 Server のメッセージオブジェクト

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. キーの名称は 'timestamp_key' で変更可能.
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

55.3.4 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' に応じた値でなくてはなりません。但し書き込み処理において ’base_ratio ’と ’ratio ’を用いた数値の逆変換は行われません。

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]
}
```

第56章　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。

56.1 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。

表 1: PD Handler SW4x がサポートするノードの一覧

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

56.2 デォルトパス

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。

表 2: PD Handler SW4x に関連するファイルのデォルトパス

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_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 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。

56.3 モデリングファイルの書式

56.3.1 構文

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

56.3.2 モデルキー

モデルキーの一覧を示します。

表 3: モデルキーの一覧

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 シリーズ 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

56.3.3 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。

表 4: モデルオブジェクト

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.

56.3.4 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。

表 5: keys オブジェクト

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	値の係数に用いる 10 を低とする冪乗値の冪指数を指定する. デフォルト値は 0

56.3.5 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。

表 6: nodeInfos オブジェクト

キー	データ型	説明
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).

56.3.6 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。

表 7: inits オブジェクト

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.

56.3.7 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。

汎用

表 8: 汎用 pools オブジェクト

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.
interva	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

表 9: SW-4280-1000 Modbus RTU ノード用 pools オブジェクト

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.

56.3.8 モデル別オブジェクト

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。

表 10: SW4X_COMMON オブジェクト

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。

表 11: SW4X_COMMON の keys オブジェクト

列番	key	include	form	ratio	説明
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 **0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)
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

SW42P0_x01 オブジェクト

SW42P0_x01 オブジェクトで設定可能なブジェクトを示します。

表 12: SW42P0_x01 オブジェクト

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_x01 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。

表 13: SW42P0_x01 の keys オブジェクト

列番	key	include	form	ratio	説明
1	temperature	true		-1	温度 (℃) -79.9 〜 79.9
2	tempeStaus	true	bool		温度の測定ステータス

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。

表 14: SW4210_1202 オブジェクト

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。

表 15: SW4210_1202 の keys オブジェクト

列番	key	include	form	ratio	説明
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_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。

表 16: SW4210_1205 オブジェクト

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。

表 17: SW4210_1205 の keys オブジェクト

列番	key	include	form	ratio	説明
1	illuminance	true		0	照度 (lx) 0 〜 99999
2	illumStatus	true	bool		照度の測定ステータス

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。

表 18: SW4210_1204 オブジェクト

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。

表 19: SW4210_1204 の keys オブジェクト

列番	key	include	form	ratio	説明
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

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。

表 20: SW4220_1000 オブジェクト

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。

表 21: SW4220_1000 の keys オブジェクト

列番	key	include	form	ratio	説明
1	humanCount	true		0	人感検出回数 0 〜 4095
2	detectionMaxWidth	true		1	人感検出幅 MAX 0 〜 255
3	detectionMinWidth	true		1	人感検出幅 MIN 0 〜 255

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。

表 22: SW4240_1000 オブジェクト

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。

表 23: SW4240_1000 の keys オブジェクト

列番	key	include	form	ratio	説明
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

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。

表 24: SW4220_1020 オブジェクト

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。

表 25: SW4220_1020 の keys オブジェクト

列番	key	include	form	ratio	説明
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

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。

表 26: SW42J0_1202 オブジェクト

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。

表 27: SW42J0_1202 の keys オブジェクト

列番	key	include	form	ratio	説明
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.

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。

表 28: SW42K0_1000 オブジェクト

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。

表 29: SW42K0_1000 の keys オブジェクト

列番	key	include	form	ratio	説明
1	integralPowerConsum	true		-4	積算電力 (Kwh) 0.0000 〜 99999999.9999
2	eepromStatus	true	bool		EEPROM ステータス

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。

表 30: SW42DD_1000 オブジェクト

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。

表 31: SW42DD_1000 の keys オブジェクト

列番	key	include	form	ratio	説明
1	distributionSystem12	true			CH1,2 の配電方式. **1P2W(1CT):** 単相二線式 1CT **3P3W(1CT):** 三相三線式 1CT **1P3W(2CT):** 単相三線式 2CT **3P3W(2CT):** 三相三線式 2CT
2	distributionSystem34	true			CH3,4 の配電方式. 前項と同じ
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) 0.000 〜 9.999
8	meanCurrent2	true		-3	CH2 平均電流 (A) 0.000 〜 9.999
9	meanCurrent3	true		-3	CH3 平均電流 (A) 0.000 〜 9.999
10	meanCurrent4	true		-3	CH4 平均電流 (A) 0.000 〜 9.999
11	instantCurrent1	true		-3	CH1 瞬時電流 (A) 0.000 〜 9.999
12	instantCurrent2	true		-3	CH2 瞬時電流 (A) 0.000 〜 9.999
13	instantCurrent3	true		-3	CH3 瞬時電流 (A) 0.000 〜 9.999
14	instantCurrent4	true		-3	CH4 瞬時電流 (A) 0.000 〜 9.999
15	meanPower1	true		-3	CH1 平均電力 (Kw) 0.000 〜 9.999
16	meanPower2	true		-3	CH2 平均電力 (Kw) 0.000 〜 9.999
17	meanPower3	true		-3	CH3 平均電力 (Kw) 0.000 〜 9.999
18	meanPower4	true		-3	CH4 平均電力 (Kw) 0.000 〜 9.999
19	instantPower1	true		-3	CH1 瞬時電力 (Kw) 0.000 〜 9.999
20	instantPower2	true		-3	CH2 瞬時電力 (Kw) 0.000 〜 9.999
21	instantPower3	true		-3	CH3 瞬時電力 (Kw) 0.000 〜 9.999
22	instantPower4	true		-3	CH4 瞬時電力 (Kw) 0.000 〜 9.999
23	integralPowerConsum1	true		-2	CH1 積算有効電力量 (Kwh) 0.00 〜 9999999.99
24	integralPowerConsum2	true		-2	CH2 積算有効電力量 (Kwh) 0.00 〜 9999999.99
25	integralPowerConsum3	true		-2	CH3 積算有効電力量 (Kwh) 0.00 〜 9999999.99
26	integralPowerConsum4	true		-2	CH4 積算有効電力量 (Kwh) 0.00 〜 9999999.99

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。

表 32: SW42DD_1100 オブジェクト

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。

表 33: SW42DD_1100 の keys オブジェクト

列番	key	include	form	ratio	説明
1	distributionSystem12	true			CH1,2 の配電方式. **1P2W(1CT):** 単相二線式 1CT **3P3W(1CT):** 三相三線式 1CT **1P3W(2CT):** 単相三線式 2CT **3P3W(2CT):** 三相三線式 2CT
2	distributionSystem34	true			CH3,4 の配電方式. 前項と同じ
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) 0.00 〜 999.9
15	meanPower1	true		-2	CH1 平均電力 (Kw) 0.00 〜 99.99
16	meanPower2	true		-2	CH2 平均電力 (Kw) 0.00 〜 99.99
17	meanPower3	true		-2	CH3 平均電力 (Kw) 0.00 〜 99.99
18	meanPower4	true		-2	CH4 平均電力 (Kw) 0.00 〜 99.99
19	instantPower1	true		-2	CH1 瞬時電力 (Kw) 0.00 〜 99.99
20	instantPower2	true		-2	CH2 瞬時電力 (Kw) 0.00 〜 99.99
21	instantPower3	true		-2	CH3 瞬時電力 (Kw) 0.00 〜 99.99
22	instantPower4	true		-2	CH4 瞬時電力 (Kw) 0.00 〜 99.99
23	integralPowerConsum1	true		-2	CH1 積算有効電力量 (Kwh) 0.00 〜 9999999.99
24	integralPowerConsum2	true		-2	CH2 積算有効電力量 (Kwh) 0.00 〜 9999999.99
25	integralPowerConsum3	true		-2	CH3 積算有効電力量 (Kwh) 0.00 〜 9999999.99
26	integralPowerConsum4	true		-2	CH4 積算有効電力量 (Kwh) 0.00 〜 9999999.99

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。

表 34: SW42DD_1000 オブジェクト

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。

表 35: SW42D0_1000 の keys オブジェクト

列番	key	include	form	ratio	説明
1	channel1LowCurrent	true	bool		CH1 の電流が 1A 未満
2	distributionSystem12	true			CH1,2 の配電方式. **1P2W(1CT):** 単相二線式 1CT **3P3W(1CT):** 三相三線式 1CT **1P3W(2CT):** 単相三線式 2CT **3P3W(2CT):** 三相三線式 2CT
3	distributionSystem34	true			CH3,4 の配電方式. 前項と同じ
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 〜 99999999.9
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

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。

表 36: SW4230_1000 オブジェクト

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。

表 37: SW4230_1000 の keys オブジェクト

列番	key	include	form	ratio	説明
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

SW4260_1x00 オブジェクト

SW4260_1x00 オブジェクトで設定可能なブジェクトを示します。

表 38: SW4260_1x00 オブジェクト

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_1x00 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。

表 39: SW4260_1x00 の keys オブジェクト

列番	key	include	form	ratio	説明
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
15	sensorConnected	true			接続センサー番号 0xNNNNNNNN

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。

表 40: SW4280_1000 オブジェクト

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	SW-4280-1000 Modbus RTU ノード用 polls オブジェクト
polls	JSON 配列	
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。

表 41: SW4280_1000 の keys オブジェクト

列番	key	include	form	ratio	説明
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 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

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。

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。

表 42: SW4260_1x20 オブジェクト

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。

表 43: SW4260_1x20 の keys オブジェクト

列番	key	include	form	ratio	説明
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
18	sensorConnected	true			接続回線番号 0xNNNNNNNN

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。

表 44: SW4000_1000, SW4100_1000, SB4020_1000, SW4300_1000, SW4500_1000 オブジェクト

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	

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。

表 45: SW4000_1000, SW4100_1000, SB4020_1000, SW4300_1000, SW4500_1000 の keys オブジェクト

列番	key	include	form	ratio	説明
1	periodic	true	bool		定期送信を示すフラグ

56.4 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": [
            <inits オブジェクト>,
            <inits オブジェクト>
    ],
    "polls": [
        <SW-4280-1000 Modbus RTU ノード用の polls オブジェクト>,
        <SW-4280-1000 Modbus RTU ノード用の polls オブジェクト>
    ]
}
```

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。

56.5 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。

56.6 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 キーに設定される「取得時間間隔」と base_time キーに設定される「基準時刻」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 SW4x 内で次のように扱われます。

表 46: 基準時刻制御における取得時間間隔の設定と実動作値

取得時間間隔の設定値	実動作値
0 〜 199	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」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。

56.7 設定ファイルの書式

56.7.1 構文

```
{
    "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>",
    "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.”
}
```

56.7.2 設定オブジェクト

表 47: PD Handler sw4x の設定 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)
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	制御メッセージ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/<localname>.sock' が設定されます.
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. 空の場合は、デフォルト値 '@/pd_repeater/<localname>.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 (下流方向制御を行う) 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	SW4x 通信モジュールを接続するシリアルデバイスのデバイス名. 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.

56.8 メッセージオブジェクト

56.8.1 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。

表 48: コマンド入力オブジェクト

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' で埋められる. 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

56.8.2 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。

表 49: 応答メッセージオブジェクト

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

56.9 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": "0x200102000000000000001234"
}
```

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
}
```

56.10 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
'0xNNNNNNNNNNNNNNNNNNNNNNNNNN' 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です。

第XIV部

PD Handler JSON フォーマット一覧

第57章　PD Handler BLE (Node.js)

57.1 Beacon

57.1.1 ビーコン送受信設定

■ データサンプル　：

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

表 1: Beacon (ビーコン送受信設定)

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

57.1.2 BLE デバイス情報送信設定

■ データサンプル

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

表 2: Beacon (BLE デバイス情報送信設定)

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

57.2 Sensor

57.2.1 TI Sensor

BLE デバイス情報送信設定

■ データサンプル

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

表 3: TI Sensor (BLE デバイス情報送信設定)

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから ':' を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	humidity	湿度	Double	△	[%]
4	temperature	温度	Double	△	[℃]
5	accelX	X 方向加速度	Double	△	[G]
6	accelY	Y 方向加速度	Double	△	[G]
7	accelZ	Z 方向加速度	Double	△	[G]
8	gyroX	X 方向角速度	Double	△	[°/s]
9	gyroY	Y 方向角速度	Double	△	[°/s]
10	gyroZ	Z 方向角速度	Double	△	[°/s]
11	magX	X 方向地磁気	Double	△	[μ T]
12	magY	Y 方向地磁気	Double	△	[μ T]
13	magZ	Z 方向地磁気	Double	△	[μ T]
14	pressure	気圧	Double	△	[hPa]
15	objectTemp	物体温度	Double	△	[℃]
16	ambientTemp	周辺温度	Double	△	[℃]
17	lux	照度	Double	△	[lux]
18	memo	メモ	String		WEB UI から設定された値

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

57.2.2 Fujitsu Sensor

ビーコン送受信設定

■ データサンプル

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

表 4: Fujitsu Sensor (ビーコン送受信設定)

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

BLE デバイス情報送信設定

■ データサンプル (接続モード：ビーコンモード)

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

表 5: Fujitsu Sensor (接続モード：ビーコンモード)

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

57.2.3 ALPS IoT Smart Module

ビーコン送受信設定

■ データサンプル (ビーコンモード：環境系フォーマット)

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

表 6: ALPS IoT Smart Module (ビーコンモード：環境系フォーマット)

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

■ データサンプル (ビーコンモード：モーション系フォーマット)

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

表 7: ALPS IoT Smart Module (ビーコンモード：モーション系フォーマット)

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

BLE デバイス情報送信設定

■ データサンプル (接続モード：データパケット 1)

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

表 8: ALPS IoT Smart Module (接続モード：データパケット 1)

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

■ データサンプル (接続モード：データパケット 2)

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

表 9: ALPS IoT Smart Module (接続モード：データパケット 2)

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから ':' を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	dataIndex	データインデックス	Integer	○	0〜255(シーケンス番号)
4	pressure	気圧	Double		[hPa]
13	humidity	湿度	Double		[%]
5	temperature	温度	Double		[℃]
6	uv	紫外線	Double		[mW/cm2]
7	ambientLight	照度	Double		[lux]
8	ms	ミリ秒	Integer	○	
9	second	秒	Integer	○	
10	minute	分	Integer	○	
11	hour	時	Integer	○	
12	memo	メモ	String		WEB UI から設定された値

■ データサンプル (ビーコンモード：環境系フォーマット)

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

表 10: ALPS IoT Smart Module (ビーコンモード：環境系フォーマット)

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

■ データサンプル (ビーコンモード：モーション系フォーマット)

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

表 11: ALPS IoT Smart Module (ビーコンモード：モーション系フォーマット)

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

57.2.4 オムロン 環境センサ

ビーコン送受信設定

■ データサンプル (ビーコンモード：IM)

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

表 12: オムロン 環境センサ (ビーコンモード：IM)

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

■ データサンプル (ビーコンモード：EP)

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

表 13: オムロン 環境センサ (ビーコンモード：EP)

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

BLE デバイス情報送信設定

■ データサンプル (接続モード)

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

表 14: オムロン 環境センサ (接続モード)

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

■ データサンプル (ビーコンモード：IM)

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

表 15: オムロン 環境センサ (ビーコンモード：IM)

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

■ データサンプル (ビーコンモード：EP)


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


表 16: オムロン 環境センサ (ビーコンモード：EP)

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

57.2.5 オムロン USB 型環境センサ

ビーコン送受信設定

■ データサンプル (ビーコンモード：Sensor data)

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

表 17: オムロン USB 型環境センサ (ビーコンモード：Sensor data)

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

BLE デバイス情報送信設定

■ データサンプル (ビーコンモード：Sensor Data)

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

表 18: オムロン USB 型環境センサ (ビーコンモード：Sensor Data)

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

57.2.6 ユニ電子 BLE 温湿度センサー (Logtta)

ビーコン送受信設定

■ データサンプル (ビーコンモード)

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

表 19: ユニ電子 BLE 温湿度センサー (ビーコンモード)

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

BLE デバイス情報送信設定

■ データサンプル (ビーコンモード)

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

表 20: ユニ電子 BLE 温湿度センサー (ビーコンモード)

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

57.2.7 ユニ電子 BLE　CO2 センサー (Logtta CO2)

ビーコン送受信設定

■ データサンプル (ビーコンモード)

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

表 21: ユニ電子 BLE CO2 センサー (ビーコンモード)

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

BLE デバイス情報送信設定

■ データサンプル (ビーコンモード)

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

表 22: ユニ電子 BLE CO2 センサー (ビーコンモード)

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

57.2.8 ユニ電子 BLE　水温センサー (Logtta Water)

ビーコン送受信設定

■ データサンプル (ビーコンモード)

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

表 23: ユニ電子 BLE 水温センサー (ビーコンモード)

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

BLE デバイス情報送信設定

■ データサンプル (ビーコンモード)

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

表 24: ユニ電子 BLE 水温センサー (ビーコンモード)

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

57.2.9 ラトックシステム Bluetooth ホコリセンサー

BLE デバイス情報送信設定

■ データサンプル

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

表 25: ラトックシステム Bluetooth ホコリセンサー

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

57.2.10 ラトックシステム Bluetooth エアクオリティモニター

BLE デバイス情報送信設定

■ データサンプル

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

表 26: ラトックシステム Bluetooth エアクオリティモニター

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

57.2.11 ラトックシステム ワットチェッカー

BLE デバイス情報送信設定

■ データサンプル

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

表 27: ラトックシステム ワットチェッカー

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

57.2.12 エレックス工業 μ PRISM

BLE デバイス情報送信設定

■ データサンプル


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

表 28: エレックス工業 μ PRISM

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

57.2.13 EnOcean マルチセンサ (EMSIB)

ビーコン送受信設定

■ データサンプル (Default)

```
{
    "time": "2020-05-19T16:10:36.425+09:00",
    "deviceId": "e500ec09abcd",
    "rssi": -82,
    "sequence": 16649,
    "temperature": 26.9,
    "humidity": 52.5,
    "ill_sensor": 216,
    "accelS": 1,
    "accelX": -0.06,
    "accelY": 0.04,
    "accelZ": 1.01,
    "contact": "Open",
    "energy_level": 90
}
```

表 29: EnOcean マルチセンサ (Default)

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから ':' を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。('in' または 'out')
9	sequence	シーケンス番号	Integer	○	
10	temperature	温度	Double		[℃]
11	humidity	湿度	Double		[%]
12	ill_sensor	照度 (センサ)	Integer		[lux]
13	accelS	加速度ステータス	Integer		
14	accelX	X 方向加速度	Double		[G]
15	accelY	Y 方向加速度	Double		[G]
16	accelZ	Z 方向加速度	Double		[G]
17	contact	マグネットコンタクト	String		"Open", "Closed", "Unknown"
18	energy_level	エネルギーレベル	Double		[%]
19	voltage	電圧	Double		[mV]
20	ill_solar	照度 (太陽電池)	Integer		[lux]
21	optional	オプショナルデータ	String		
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

BLE デバイス情報送信設定

■ データサンプル (Default)

```
{
    "time": "2020-05-19T16:12:25.562+09:00",
    "memo": "EnOcean EMSIB",
    "deviceId": "e500ec09abcd",
    "sequence": 16532,
    "temperature": 26.9,
    "humidity": 52.5,
    "ill_sensor": 216,
    "accelS": 1,
    "accelX": -0.06,
    "accelY": 0.04,
    "accelZ": 1.01,
    "contact": "Open",
    "energy_level": 90
}
```

表 30: EnOcean マルチセンサ (Default)

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから ':' を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	sequence	シーケンス番号	Integer	○	
4	temperature	温度	Double	○	[℃]
5	humidity	湿度	Double	○	[%]
6	ill_sensor	照度 (センサ)	Integer		[lux]
7	accelS	加速度ステータス	Integer		
8	accelX	X 方向加速度	Double		[G]
9	accelY	Y 方向加速度	Double		[G]
10	accelZ	Z 方向加速度	Double		[G]
11	contact	マグネットコンタクト	String		"Open", "Closed", "Unknown"
12	energy_level	エネルギーレベル	Double		[%]
13	voltage	電圧	Double		[mV]
14	ill_solar	照度 (太陽電池)	Integer		[lux]
15	optional	オプショナルデータ	String		
16	memo	メモ	String		WEB UI から設定された値

57.2.14 EnOcean 照度センサ&人感センサ (EMDCB)

ビーコン送受信設定

■ データサンプル (Default)

```
{
    "time": "2020-05-19T17:12:59.237+09:00",
    "deviceId": "e50000000252",
    "rssi": -85,
    "sequence": 55370,
    "energy_level": 90,
    "ill_solar": 216,
    "ill_sensor": 216,
    "occupancy": ""Occupied
}
```

表 31: EnOcean 照度センサ&人感センサ (Default)

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから ':' を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。('in' または 'out')
9	sequence	シーケンス番号	Integer	○	
10	ill_sensor	照度 (センサ)	Integer		[lux]
11	occupancy	人感センサ	String		"Not occupied", "Occupied", "Unknown"
12	energy_level	エネルギーレベル	Double		[%]
13	voltage	電圧	Double		[mV]
14	ill_solar	照度 (太陽電池)	Integer		[lux]
15	optional	オプショナルデータ	String		
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

BLE デバイス情報送信設定

■ データサンプル (Default)

```
{
    "time": "2020-05-19T16:12:25.562+09:00",
    "memo": "EnOcean EMDCB",
    "deviceId": "e500ec09abcd",
    "sequence": 55370,
    "energy_level": 90,
    "ill_solar": 216,
    "ill_sensor": 216,
    "occupancy": ""Occupied
}
```

表 32: EnOcean 照度センサ&人感センサ (Default)

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから ':' を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	sequence	シーケンス番号	Integer	○	
4	ill_sensor	照度 (センサ)	Integer		[lux]
5	occupancy	人感センサ	String		"Not occupied", "Occupied", "Unknown"
6	energy_level	エネルギーレベル	Double		[%]
7	voltage	電圧	Double		[mV]
8	ill_solar	照度 (太陽電池)	Integer		[lux]
9	optional	オプショナルデータ	String		
10	memo	メモ	String		WEB UI から設定された値

第58章　PD Handler BLE (C) with Lua

注釈:

- ファームウェア 1.x、2.x は、Lua に対応しておりません。
- ログに出力される JSON データは順不同です。

58.1 Beacon

PD Handler BLE (Node.js) の *Beacon* を参照して下さい。

58.2 Sensor

接続モードのセンサには対応していません。

対応センサは *PD Handler BLE (Node.js)* の *Sensor* のビーコンモードを参照ください。
また、BLE デバイス情報送信設定の場合にも rssi 情報が付与されます。
以下、*PD Handler BLE (C) with Lua* で追加のデバイスのみ記載します。

58.2.1 ナカヨ 呼出しボタン

ビーコン送受信設定

■ データサンプル

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

表 1: ナカヨ 呼出しボタン (ビーコン送受信設定)

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

BLE デバイス情報送信設定

■ データサンプル

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

表 2: ナカヨ 呼出しボタン (BLE デバイス情報送信設定)

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

第59章　PD Handler UART

注釈: ログに出力される JSON データは順不同です。

59.1 EnOcean with Lua

注釈: ファームウェア 1.x、2.x は、Lua に対応しておりません。

59.1.1 人感センサー (EEP : A50701)

■ データサンプル

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

表 1: 人感センサー (EEP : A50701)

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

59.1.2 あけしめセンサー (EEP : D50001)

■ データサンプル

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

表 2: あけしめセンサー (EEP : D50001)

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

59.1.3 温度センサー (EEP : A50205)

■ データサンプル

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

表 3: 温度センサー (EEP : A50205)

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

59.1.4 温湿度センサー (EEP : A50402)

■ データサンプル

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

表 4: 温湿度センサー (EEP : A50402)

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

59.1.5 温湿度センサー (EEP : A50403)

■ データサンプル

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

表 5: 温湿度センサー (EEP : A50403)

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

59.1.6 2 相式 CT センサー (EEP : A51201)

■ データサンプル

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

表 6: 2 相式 CT センサー (EEP : A51201)

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

59.1.7 3 相式 CT センサー (EEP : D23202)

■ データサンプル

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

表 7: 3 相式 CT センサー (EEP : D23202)

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

59.1.8 大気圧センサー (EEP : A50501)

■ データサンプル

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

表 8: 大気圧センサー (EEP : A50501)

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

59.1.9 照度センサー (EEP : A50601)

■ データサンプル

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

表 9: 照度センサー (EEP : A50601)

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

59.1.10 照度センサー (EEP : A50602)

■ データサンプル

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

表 10: 照度センサー (EEP : A50602)

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

59.1.11 照度センサー (EEP : A50605)

■ データサンプル

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

表 11: 照度センサー (EEP : A50605)

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

59.1.12 CO2 センサー (EEP : A50904)

■ データサンプル

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

表 12: CO2 センサー (EEP : A50904)

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

59.1.13 デジタル入力センサー (EEP : A53005)

■ データサンプル

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

表 13: デジタル入力センサー (EEP : A53005)

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

59.1.14 2 ロッカースイッチ (EEP : F60204)

■ データサンプル

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

表 14: 2 ロッカースイッチ (EEP : F60204)

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

59.1.15 マルチセンサ (EEP : D21441)

■ データサンプル

```
{
    "deviceId": "04117EAE",
    "time": "2020-06-10T10:45:09.625+09:00",
    "temperature": 24.25,
    "humidity": 50.51,
    "ill": 153,
    "accelS": 0,
    "accelX": 0.01,
    "accelY": -0.02,
    "accelZ": 1.01,
    "contact": 0,
    "EEP": "D21441",
    "memo": "multi function sensors",
    "rssi": -81
}
```

表 15: マルチセンサ (EEP : D21441)

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
4	rssi	受信信号強度	Integer	○	
5	memo	メモ	String		WEB UI にて設定した値
6	temperature	温度	Double	○	[℃]
7	humidity	湿度	Double	○	[%]
8	accelS	加速度ステータス	Integer	○	
9	accelX	X 方向加速度	Double	○	[G]
10	accelY	Y 方向加速度	Double	○	[G]
11	accelZ	Z 方向加速度	Double	○	[G]
12	contact	マグネットコンタクト	Integer	○	0: Open, 1: Closed

59.1.16 マルチセンサ (EEP : D21440)

■ データサンプル

```
{
    "deviceId": "04117EAE",
    "time": "2020-06-10T10:45:09.625+09:00",
    "temperature": 24.25,
    "humidity": 50.51,
    "ill": 153,
    "accelS": 0,
    "accelX": 0.01,
    "accelY": -0.02,
    "accelZ": 1.01,
    "EEP": "D21440",
    "memo": "multi function sensors",
    "rssi": -81
}
```

表 16: マルチセンサ (EEP : D21440)

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
4	rssi	受信信号強度	Integer	○	
5	memo	メモ	String		WEB UI にて設定した値
6	temperature	温度	Double	○	[℃]
7	humidity	湿度	Double	○	[%]
8	accelS	加速度ステータス	Integer	○	
9	accelX	X 方向加速度	Double	○	[G]
10	accelY	Y 方向加速度	Double	○	[G]
11	accelZ	Z 方向加速度	Double	○	[G]

59.1.17 温湿度センサー (EEP : A50401)

■ データサンプル

```
{
    "deviceId": "04117EAE",
    "time": "2020-06-10T10:45:09.625+09:00",
    "temperature": 24.25,
    "humidity": 50.51,
    "EEP": "A50401",
    "memo": "Temperature and Humidity Sensor",
    "rssi": -66
}
```

表 17: 温湿度センサー (EEP : A50401)

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

59.1.18 照度センサー (EEP : A50603)

■ データサンプル

```
{
    "deviceId": "04117EAE",
    "time": "2020-06-10T10:45:09.625+09:00",
    "svc": 2.7800,
    "ill": 260,
    "EEP": "A50603",
    "memo": "ill",
    "rssi": -65
}
```

表 18: 照度センサー (EEP : A50603)

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

59.1.19 マルチセンサ (EEP : A51405)

■ データサンプル

```
{
    "deviceId": "04117EAE",
    "time": "2020-06-10T10:45:09.625+09:00",
    "svc": 2.7800,
    "vib": 0,
    "EEP": "A51405",
    "memo": "multi-func sensor",
    "rssi": -65
}
```

表 19: マルチセンサ (EEP : A51405)

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
4	rssi	受信信号強度	Integer	○	
5	memo	メモ	String		WEB UI にて設定した値
6	svc	供給電圧	Double	○	[V]
7	vib	振動	Integer	○	0: No vibration detected, 1: Vibration detected

59.1.20 RAW データ時

■ データサンプル

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

表 20: RAW データ時

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

59.2 3.2. Wi-SUN

59.2.1 瞬時電力 (B ルート)

■ データサンプル

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

表 21: 瞬時電力 (B ルート)

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

59.2.2 積算電力 (B ルート)

■ データサンプル

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

表 22: 積算電力 (B ルート)

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

計算方式

- 正方向 積算電力

  <Total> = <cumu> × <ratio> × <unit>

- 逆方向 積算電力

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

第60章　PD Handler HVSMC

注釈: ログに出力される JSON データは順不同です。

60.1 RAW データ

■ データサンプル

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

表 1: RAW データ

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

60.2 動作状態

■ データサンプル

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

表 2: 動作状態

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

60.3 異常発生状態

■ データサンプル

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

表 3: 異常発生状態

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

60.4 ECHONET Lite 属性情報

■ データサンプル


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


表 4: ECHONET Lite 属性情報

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

60.5 高圧スマート電力量メータ属性情報

■ データサンプル

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

表 5: 高圧スマート電力量メータ属性情報

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

60.6 定時計測値

■ データサンプル

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

表 6: 定時計測値

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

60.7 計測値

■ データサンプル

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

表 7: 計測値

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

60.8 需要電力

■ データサンプル

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

表 8: 需要電力

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

第61章　PD Handler Modbus

注釈:

- 出力される JSON データは順不同です。
- 接続プロトコル（TCP と RTC）とデータ形式（レジスタとデジタル）の組み合わせにより、データ、リクエストメッセージ、応答メッセージの各サンプルは本来４通りとなりますが、TCP プロトコルによるレジスタ入出力と RTU プロトコルによるビット入出力のみ掲載しています。

61.1 Modbus クライアント (Modbus マスター)

61.1.1 PLC 機器へのポーリング動作時

TCP プロトコルによるレジスタ出力又はレジスタ入力の読み込み

■ データサンプル

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

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

RTU プロトコルによるデジタル出力又はデジタル入力の読み込み

■ データサンプル

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

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

61.1.2 クラウドからのオンデマンド動作時

TCP プロトコルによるレジスタ出力又はレジスタ入力の読み込み

■ リクエストメッセージサンプル

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

#	JSON キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	'tcp'.
2	node	取得先 IP アドレス	String	○	PLC 機器の IP アドレス
3	port	ポート番号	Integer	○	
4	unit	Modbus Unit ID	Integer		省略時は 255
5	address	読み込みアドレス	Integer *1		省略時は 0
6	function	Modbus function code	integer *1	○	**3:** read holding registers **4:** read input registers
7	number	読み込むレジスタ数	Integer *1		省略時は 1
8	data_type	データの型	String		**uint16_t :** 符号なし 16bits **int16_t :** 符号付 16bits **uint32lsb_t :** 符号なし 32bits LSB **uint32msb_t :** 符号なし 32bits MSB **int32lsb_t :** 符号付 32bits LSB **int32msb_t :** 符号付 32bits MSB 省略時は、uint16_t

注釈:

1. String 型 '0x' から始まる 16 進数表記も可能です。

■ 応答メッセージサンプル

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

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

RTU プロトコルによるデジタル出力又はデジタル入力の読み込み

■ リクエストメッセージサンプル

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

#	JSON キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	'rtu'.
2	device	デバイスファイル名	String	○	PLC 機器を接続するシリアルポート
3	unit	Modbus Unit ID	Integer	○	
4	address	読み込みアドレス	Integer *1		省略時は 0
5	function	Modbus function code	integer *1	○	**1:** read coils **2:** read discrete inputs
6	number	読み込むビット数	Integer *1		省略時は 1

注釈:

1. String 型 '0x' から始まる 16 進数表記も可能です。

■ 応答メッセージサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は 'done'
4	protocol	プロトコル	String	○	リクエストされた値. 'rtu'
5	device	デバイスファイル名	String	○	リクエストされた値
6	unit	Modbus Unit ID	Integer	○	リクエストされた値
7	memo	メモ	String		WEB UI から設定さた値
8	address	読み込みアドレス	Integer	○	リクエストされた値
9	function	Modbus function code	integer	○	リクエストされた値
10	values	読み込み値	Integer 配列	○	0 又は 1. 配列数は number キーでリクエストされたレジスタ数に応じて可変.

TCP プロトコルによるレジスタ入力への書き込み

■ リクエストメッセージサンプル

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

表 1: TCP プロトコルによるレジスタ入力への書き込み

#	JSON キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	'tcp'.
2	node	取得先 IP アドレス	String	○	PLC 機器の IP アドレス
3	port	ポート番号	Integer	○	
4	unit	Modbus Unit ID	Integer		省略時は 255
5	address	書き込みアドレス	Integer *1		省略時は 0
6	function	Modbus function code	integer *1	○	**6:** write single register **16:** write multiple registers **23:** write and read registers
7	data_type	データの型	String		**uint16_t :** 符号なし 16bits **int16_t :** 符号付 16bits **uint32lsb_t :** 符号なし 32bits LSB **uint32msb_t :** 符号なし 32bits MSB **int32lsb_t :** 符号付 32bits LSB **int32msb_t :** 符号付 32bits MSB 省略時は、uint16_t
8	values	書き込む値	Integer 配列	○	function キーが 6 の場合、は先頭の 1 レジスタを書き込む.

注釈:

1. String 型 '0x' から始まる 16 進数表記も可能です。

■ 応答メッセージサンプル

```
{
    "time":  "2017-02-03T14:44:37.020+09:00",
    "reply_to":  "73771103b4765ed0ce859ac912321c04",
    "result":  "done"
    "protocol":  "tcp",
    "node":  "172.16.7.250",
    "port": 1502,
    "unit": 255,
    "address": 171,
    "function": 16,
    "data_type":  "uint32lsb_t",
    "values": [ 42949672951,  21474836471]
}
```

#	JSON キー	内容	データ型	常駐	補足
1	time	データ書き込み日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は 'done'
4	protocol	プロトコル	String	○	リクエストされた値. 'tcp'
5	node	取得先 IP アドレス	String	○	リクエストされた値
6	port	ポート番号	Integer	○	リクエストされた値
7	unit	Modbus Unit ID	Integer	○	リクエストされた値
9	memo	メモ	String		WEB UI から設定さた値
10	address	書き込みアドレス	Integer	○	リクエストされた値
11	function	Modbus function code	integer	○	リクエストされた値
12	data_type	データの型	String	○	リクエストされた値
13	values	書き込んだ値	Integer 配列	○	リクエストされた値

RTU プロトコルによるデジタル入力への書き込み

■ リクエストメッセージサンプル

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

#	JSON キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	'rtu'.
2	device	デバイスファイル名	String	○	PLC 機器を接続するシリアルポート
3	unit	Modbus Unit ID	Integer	○	
4	address	書き込みアドレス	Integer *1		省略時は 0
5	function	Modbus function code	integer *1	○	**5:** write single coil **15:** write multiple coils
7	values	書き込む値	Integer 配列	○	0 又 は 1. function キーが 5 の場合、は先頭の 1 ビットを書き込む.

注釈:

1. String 型 '0x' から始まる 16 進数表記も可能です。

■ 応答メッセージサンプル

```
{
    "time": "2017-02-03T14:44:37.020+09:00",
    "reply_to": "0408f69db38b4d89f25d026d6d9449b7",
    "result": "done"
    "protocol": "rtu",
    "device": "/dev/ttyRS485",
    "unit": 21,
    "address": 206,
    "function": 15,
    "values": [ 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1]
}
```

#	JSON キー	内容	データ型	常駐	補足
1	time	データ書き込み日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は 'done'
4	protocol	プロトコル	String	○	リクエストされた値. 'rtu'
5	device	デバイスファイル名	String	○	リクエストされた値
6	unit	Modbus Unit ID	Integer	○	リクエストされた値
7	memo	メモ	String		WEB UI から設定さた値
8	address	書き込みアドレス	Integer	○	リクエストされた値
9	function	Modbus function code	integer	○	リクエストされた値
10	values	書き込んだ値	Integer 配列	○	リクエストされた値

スレーブ ID の読み出し

■ リクエストメッセージサンプル

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

#	JSON キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	'rtu'.
2	device	デバイスファイル名	String	○	PLC 機器を接続するシリアルポート
3	unit	Modbus Unit ID	Integer	○	
4	function	Modbus function code	integer *1	○	**17:** report slave id

注釈:

1. String 型 '0x' から始まる 16 進数表記も可能です。

■応答メッセージサンプル

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

#	JSON キー	内容	データ型	必須	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は 'done'
4	protocol	プロトコル	String	○	リクエストされた値. 'rtu'
5	device	デバイスファイル名	String	○	リクエストされた値
6	unit	Modbus Unit ID	Integer	○	WEB UI から設定さた値
7	memo	メモ	String		WEB UI から設定さた値
8	function	Modbus function code	integer	○	リクエストされた値 17
9	values	接続されている Modbus Unit ID の一覧	Integer 配列	○	

エラー時

■ 応答メッセージサンプル

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

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

61.2 Modbus サーバー (Modbus スレーブ)

61.2.1 PLC 機器からの書き込み動作時

TCP プロトコルによるレジスタ入力への書き込み

■ データサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	time	データ書き込み日時	String	○	ISO8601 拡張書式
2	protocol	プロトコル	String	○	WEB UI から設定された値.'tcp'.
3	node	書き込み元 IP アドレス	String	○	WEB UI から設定された値.
4	port	ポート番号	Integer	○	502 (固定)
5	unit	Modbus Unit ID	Integer	○	255 (固定)
6	memo	メモ	String	○	WEB UI から設定された値
7	address	書き込みアドレス	Integer	○	0 〜 2048 (registers) の範囲
8	function	Modbus function code	integer	○	**6:** write single register **16:** write multiple registes
9	values	書き込まれた値	Integer 配列	○	16 ビット符号無し整数値. 配列数は書き込まれたレジスタ数に応じて可変.

RTU プロトコルによるデジタル入力への書き込み

■ データサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	time	データ書き込み日時	String	○	ISO8601 拡張書式
2	protocol	プロトコル	String	○	WEB UI から設定された値.'rtu'.
3	device	デバイスファイル名	String	○	WEB UI から設定された値.
4	unit	Modbus Unit ID	Integer	○	WEB UI から設定された値.
5	memo	メモ	String	○	WEB UI から設定された値
6	address	書き込みアドレス	Integer	○	0 〜 2048 (registers) の範囲
7	function	Modbus function code	integer	○	**5:** write single coil **15:** write multiple coils
9	values	書き込まれた値	Integer 配列	○	0 または 1. 配列数は書き込まれたビット数に応じて可変.

61.2.2 クラウドからのオンデマンド動作時

レジスタ出力又はレジスタ入力の読み込み

■ リクエストメッセージサンプル

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

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

注釈:

1. String 型 '0x' から始まる 16 進数表記も可能です。

■ 応答メッセージサンプル (サーバが TCP 接続設定時)

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

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

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

デジタル出力又はデジタル入力の読み込み

■ リクエストメッセージサンプル

```
{
    "function": 2,
    "address": 37,
    "number": 20,
}
```

#	JSON キー	内容	データ型	必須	補足
1	function	Modbus function code	integer *1	○	**1:** read coils **2:** read discrete inputs
2	address	読み込みアドレス	Integer *1		省略時は 0
3	number	読み込むレジスタ数	Integer *1		省略時は 1

注釈:

1. String 型 '0x' から始まる 16 進数表記も可能です。

■ 応答メッセージサンプル (サーバが RTU 接続設定時)

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

}
```

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

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

レジスタ出力又はレジスタ入力への書き込み

■ リクエストメッセージサンプル

```
{
    "function": 16,
    "address": "0x0ab",
    "data_type": "uint32lsb_t",
    "valuse": [ 42949672951, 21474836471]
}
```

#	JSON キー	内容	データ型	必須	補足
1	function	Modbus function code	integer *1	○	**6:** write single register **10:** write single input register **16:** write multiple registers **20:** write multiple input registers **23:** write and read registers
2	address	書き込みアドレス	Integer *1		省略時は 0
3	data_type	データの型	String		**uint16_t:** 符号なし 16bits **int16_t:** 符号付 16bits **uint32lsb_t:** 符号なし 32bits LSB **uint32msb_t:** 符号なし 32bits MSB **int32lsb_t:** 符号付 32bits LSB **int32msb_t:** 符号付 32bits MSB 省略時は、uint16_t
4	values	書き込む値	Integer 配列	○	function キーが 6 又は 10 の場合、は先頭の 1 レジスタを書き込む.

注釈: 1. String 型 '0x' から始まる 16 進数表記も可能です。

- function の 内、10.write single input register と 20.write multiple input registes は、本来の Modubus プロトコルには存在しない機能です。

■ 応答メッセージサンプル (サーバが TCP 接続設定時)

```
{
    "time": "2017-02-03T14:44:37.020+09:00",
    "reply_to": " 35cf8fa6243d87e0ebb0c2aaaf8eeecf",
    "result": "done"
    "protocol": "tcp",
    "node": "172.16.7.249",
    "port": 502,
    "unit": 255,
    "address": 171,
    "function": 16,
    "data_type": "uint32lsb_t",
    "values": [ 42949672951, 21474836471]
}
```

#	JSON キー	内容	データ型	常駐	補足
1	time	データ書き込み日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は 'done'
4	protocol	プロトコル	String	○	WEB UI から設定さた値. 'tcp'
5	node	サーバ自身の IP アドレス	String	○	
6	port	ポート番号	Integer	○	502（固定）
7	unit	Modbus Unit ID	Integer	○	255（固定）
9	memo	メモ	String		WEB UI から設定さた値
10	address	書き込みアドレス	Integer	○	リクエストされた値
11	function	Modbus function code	integer	○	リクエストされた値
12	data_type	データの型	String	○	リクエストされた値
13	values	書き込んだ値	Integer 配列	○	リクエストされた値

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

デジタル出力又はデジタル入力への書き込み

■ リクエストメッセージサンプル

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

#	JSON キー	内容	データ型	必須	補足
1	function	Modbus function code	integer *1	○	**5:** write single coil **9:** write single discrete_input **15:** write multiple coils **19:** write multiple discrete input
2	address	書き込みアドレス	Integer *1		省略時は 0
3	values	書き込む値	Integer 配列	○	0 又 は 1. function キーが 5 又は 9 の場合、は先頭の 1 ビットを書き込む.

注釈:

1. String 型 '0x' から始まる 16 進数表記も可能です。

- function の内、9.write single discrete input と 19.write multiple discrete input は、本来の Modubus プロトコルには存在しない機能です。

■ 応答メッセージサンプル (サーバが RTU 接続設定時)

```
{
    "time": "2017-02-03T14:44:37.020+09:00",
    "reply_to": "c4348e30643dac56cb61bac9743729e7",
    "result": "done"
    "protocol": "rtu",
    "device": "/dev/ttyRS485",
    "unit": 21,
    "address": 206,
    "function": 15,
    "values": [ 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1]
}
```

#	JSON キー	内容	データ型	常駐	補足
1	time	データ書き込み日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は 'done'
4	protocol	プロトコル	String	○	WEB UI から設定さた値. 'rtu'
5	device	デバイスファイル名	String	○	WEB UI から設定さた値
6	unit	Modbus Unit ID	Integer	○	WEB UI から設定さた値
7	memo	メモ	String		WEB UI から設定さた値
8	address	書き込みアドレス	Integer	○	リクエストされた値
9	function	Modbus function code	integer	○	リクエストされた値
10	values	書き込んだ値	Integer 配列	○	リクエストされた値

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

スレーブ ID の読み出し

■ リクエストメッセージサンプル

```
{
    "function": 17
}
```

#	JSON キー	内容	データ型	必須	補足
1	function	Modbus function code	integer *1	○	17. report slave id

■応答メッセージサンプル

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

#	JSON キー	内容	データ型	必須	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は 'done'
4	protocol	プロトコル	String	○	WEB UI から設定さた値 'rtu'
5	device	デバイスファイル名	String	○	リクエストされた値
6	unit	Modbus Unit ID	Integer	○	WEB UI から設定さた値
7	memo	メモ	String		WEB UI から設定さた値
8	function	Modbus function code	integer	○	リクエストされた値 17
9	values	接続されている Modbus Unit ID の一覧	Integer 配列	○	

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

エラー時

■ 応答メッセージサンプル

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

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

第62章　PD Handler Modbus 2

注釈:

- 出力される JSON データは順不同です。

62.1 Modbus クライアント (Modbus マスター)

62.1.1 PLC 機器へのポーリング動作時

■ データサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式. *1
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値. *2
3	maker	メーカー名	String		WEB UI から設定された値. *3
4	product	プロダクト名	String		WEB UI から設定された値. *3
5	model	モデル名	String		WEB UI から設定された値. *3
6	sku	SKU 名	String		WEB UI から設定された値. *3

注釈: 上記以外の項目は、モデリングファイルの設定に応じて異なります。

1. timestamp_key キーにより、JSON キーを変更することが可能です。
2. id_form キーの設定に応じて、整数または '0x0 ' から始まる大文字 16 進数、小文字 16 進数の何れかとなります。
3. モデリングの設定応じて非表示とすることができます。

62.1.2 クラウドからのオンデマンド動作時

ポーリング中のレジスタに対するオンデマンド読み込み

■ リクエストメッセージサンプル

```
{
    "unitId": 1
}
```

#	JSON キー	内容	必須	常駐	補足
1	unitId	Modbus Unit ID	Integer or String	○	整数または、'0x' から始まる 16 進数文字列.

■ 応答メッセージサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式. *1
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値. *2
3	maker	メーカー名	String		WEB UI から設定された値. *3
4	product	プロダクト名	String		WEB UI から設定された値. *3
5	model	モデル名	String		WEB UI から設定された値. *3
6	sku	SKU 名	String		WEB UI から設定された値. *3
7	request_from	リクエスト元クラウド ID	String		*4
8	reply_to	リクエストメッセージの MD5 値	String		*4
9	result	取得ステータス	Boolena		通常は true. *4

注釈: 上記以外の項目は、モデリングファイルの設定に応じて異なります。

1. timestamp_key キーにより、JSON キーを変更することが可能です。
2. id_form キーの設定に応じて、整数または '0x0 ' から始まる大文字 16 進数、小文字 16 進数の何れかとなります。
3. モデリングの設定応じて非表示とすることができます。
4. reply キー が true に設定されている場合のみ表示されます。

レジスタに対するオンデマンド書き込み

■ リクエストメッセージサンプル

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

#	JSON キー	内容	必須	常駐	補足
1	unitId	Modbus Unit ID	Integer or String	○	整数または、'0x' から始まる 16 進数文字列.
2	rawRequest	RAW リクエスト	JSON obj		モデリングファイルの設定に応じて異なります。
3	write	書込み制御	JSON obj		モデリングファイルの設定に応じて異なります。

■応答メッセージサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式. *1
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値. *2
3	maker	メーカー名	String		WEB UI から設定された値. *3
4	product	プロダクト名	String		WEB UI から設定された値. *3
5	model	モデル名	String		WEB UI から設定された値. *3
6	sku	SKU 名	String		WEB UI から設定された値. *3
7	request_from	リクエスト元クラウド ID	String		*4
8	reply_to	リクエストメッセージの MD5 値	String		*4
9	result	取得ステータス	Boolena		通常は true. *4

注釈: 上記以外の項目は、モデリングファイルの設定に応じて異なります。

1. timestamp_key キーにより、JSON キーを変更することが可能です。
2. id_form キーの設定に応じて、整数または '0x0 ' から始まる大文字 16 進数、小文字 16 進数の何れかとなります。
3. モデリングの設定応じて非表示とすることができます。
4. reply キー が true に設定されている場合のみ表示されます。

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

エラー発生時の応答メッセージ

■応答メッセージサンプル

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

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

注釈: reply キー が true に設定されている場合のみ応答を返します。

62.2 Modbus サーバー (Modbus スレーブ)

62.2.1 PLC 機器からの書き込み動作時

レジスタ出力への書き込み

■ データサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ書き込み日時	String	○	ISO8601 拡張書式. *1
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値. *2
3	maker	メーカー名	String		WEB UI から設定された値. *3
4	product	プロダクト名	String		WEB UI から設定された値. *3
5	model	モデル名	String		WEB UI から設定された値. *3
6	sku	SKU 名	String		WEB UI から設定された値. *3

注釈: 上記以外の項目は、モデリングファイルの設定に応じて異なります。

1. timestamp_key キーにより、JSON キーを変更することが可能です。
2. id_form キーの設定に応じて、整数または '0x0 ' から始まる大文字 16 進数、小文字 16 進数の何れかとなります。
3. モデリングの設定応じて非表示とすることができます。

62.2.2 クラウドからのオンデマンド動作時

レジスタの読み込み

■ リクエストメッセージサンプル

```
{ }
```

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

■ 応答メッセージサンプル

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

読み込みのリクエストに対しては、全てのレジスタの情報を返します。

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式. *1
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値. *2
3	maker	メーカー名	String		WEB UI から設定された値. *3
4	product	プロダクト名	String		WEB UI から設定された値. *3
5	model	モデル名	String		WEB UI から設定された値. *3
6	sku	SKU 名	String		WEB UI から設定された値. *3
7	request_from	リクエスト元クラウド ID	String		*4
8	reply_to	リクエストメッセージの MD5 値	String		*4
9	result	取得ステータス	Boolena		通常は true. *4

注釈: 上記以外の項目は、モデリングファイルの設定に応じて異なります。

1. timestamp_key キーにより、JSON キーを変更することが可能です。
2. id_form キーの設定に応じて、整数または '0x0 ' から始まる大文字 16 進数、小文字 16 進数の何れかとなります。
3. モデリングの設定応じて非表示とすることができます。
4. reply キー が true に設定されている場合のみ表示されます。

レジスタへの書き込み

■ リクエストメッセージサンプル

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

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

■ 応答メッセージサンプル

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

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

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式. *1
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値. *2
3	maker	メーカー名	String		WEB UI から設定された値. *3
4	product	プロダクト名	String		WEB UI から設定された値. *3
5	model	モデル名	String		WEB UI から設定された値. *3
6	sku	SKU 名	String		WEB UI から設定された値. *3
7	request_from	リクエスト元クラウド ID	String		*4
8	reply_to	リクエストメッセージの MD5 値	String		*4
9	result	取得ステータス	Boolena		通常は true. *4

注釈: 上記以外の項目は、モデリングファイルの設定に応じて異なります。

1. timestamp_key キーにより、JSON キーを変更することが可能です。
2. id_form キーの設定に応じて、整数または '0x0 ' から始まる大文字 16 進数、小文字 16 進数の何れかとなります。
3. モデリングの設定応じて非表示とすることができます。
4. reply キー が true に設定されている場合のみ表示されます。

エラー発生時の応答メッセージ

■応答メッセージサンプル

```
{
    "timestamp": "2019-11-15T11:34:54.679+09:00",
    "request_from": "0x01(0)",
    "reply_to": "8473b3a74dfe726d256314fcfd1186cf",
    "result": false,
    "reason": "parser_sub_socket(): invalid write request '{\"inBitr10\": 0}'"
}
```

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

注釈:

- reply キー が true に設定されている場合のみ応答を返します。
- 複数の書き込みリクエストが、行われた場合で、そのいずれかの書き込みが成功した場合、エラー出力は行われず、正常に書き込まれたレジスタの情報のみが返されます。

第63章　PD Handler SW4x

PD Handler SW4x の出力メッセージは、モデリングファイルを定義することによりへ変更することができます。

本項では、PD Handler SW4x にハードコーディングされたデフォルトの単方向通信メッセージフォーマットを記載します。

双方向通信のフォーマットについては、コマンドに応じて変化し多義に渡るため、掲載しておりません。

双方向通信時に用いられる JSON キーについては、 **データハンドリング設定リファレンスガイド** の *PD Handler SW4x* に記載されるモデル別オブジェクトを参照してください。

「常駐」項については、モデリングファイルで定義可能であるため記載いたしません。

注釈:

出力される JSON データは順不同です。

63.1 SW42P0-1x01 温度センサーノード

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x40",
    "unitType": "0x00",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x40"
        }
    ],
    "temperature": 27.2
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x00
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	temperature	温度	Real	-79.9 〜 79.9 (℃)
7	tempeStatus	温度の測定ステータス	Boolean	測定エラー発生時のみ
8	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.2 SW4210-1202 温・湿度センサーノード

■ データサンプル


```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x41",
    "unitType": "0x01",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x41"
        }
    ],
    "temperature": 27.2,
    "humidity": 72.1
}
```


#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x01
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	temperature	温度	Real	-79.9 〜 79.9 (℃)
7	humidity	湿度	Real	0.0 〜 100.0 (%)
8	tempeStatus	温度の測定ステータス	Boolean	測定エラー発生時のみ
9	humidStatus	湿度の測定ステータス	Boolean	測定エラー発生時のみ
8	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.3 SW4210-1205 照度センサーノード

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x42",
    "unitType": "0x02",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x42"
        }
    ],
    "illuminance": 623
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x02
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	illuminance	照度	Integer	0 〜 99999 (lx)
7	illumStatus	照度の測定ステータス	Boolean	測定エラー発生時のみ
8	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.4 SW4210-1204 温・湿度・照度センサーノード

■ データサンプル


```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x43",
    "unitType": "0x03",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x43"
        }
    ],
    "temperature": 27.2,
    "humidity": 72.1,
    "illuminance": 623
}
```


#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x03
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	temperature	温度	Real	(℃) -79.9 〜 79.9 (℃)
7	humidity	湿度	Real	0.0 〜 100.0 (%)
8	illuminance	照度	Integer	0 〜 99999 (lx)
9	tempeStatus	温度の測定ステータス	Boolean	測定エラー発生時のみ
10	humidStatus	湿度の測定ステータス	Boolean	測定エラー発生時のみ
11	illumStatus	照度の測定ステータス	Boolean	測定エラー発生時のみ
12	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.5 SW4220-1000 人感センサーノード（活動量）

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x49",
    "unitType": "0x09",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x49"
        }
    ],
    "humanCount": 451,
    "detectionMaxWidth": 200,
    "detectionMinWidth": 32
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x09
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満）
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	humanCount	人感検出回数	Integer	0 〜 4095
7	detectionMaxWidth	人感検出幅 MAX	Integer	0 〜 255
8	detectionMinWidth	人感検出幅 MIN	Integer	0 〜 255
9	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.6 SW4240-1000 パルスカウントノード

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x4A",
    "unitType": "0x0A",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x4A"
        }
    ],
    "pulseCount1": 23567,
    "pulseCount2": 1327
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x0A
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	pulseCount1	パルスカウント数 1	Integer	0 〜 99999999
7	pulseCount2	パルスカウント数 2	Integer	0 〜 99999999
8	eepromStatus	EEPROM ステータス	Boolean	EEPROM R/W エラー発生時のみ
9	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.7 SW4220-1020 人感センサーノード（イベントドリブン仕様）

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x4B",
    "unitType": "0x0B",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x4B"
        }
    ],
    "humanDetectCount": 267
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x0B
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満）
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	humanDetectCount	人感検知カウント	Integer	0 〜 99999999
7	periodic	定期送信を示すフラグ	Boolean	定期送信時のみ
8	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.8 SW42J0-1202 リモコン温湿度ノード

■ データサンプル


```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x4D",
    "unitType": "0x0D",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x4D"
        }
    ],
    "temperature": 27.2,
    "humidity": 72.1
}
```


#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x0D
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	temperature	温度	Real	-79.9 〜 79.9 (℃)
7	humidity	湿度	Real	0.0 〜 100.0 (%)
8	tempeStatus	温度の測定ステータス	Boolean	測定エラー発生時のみ
9	humidStatus	湿度の測定ステータス	Boolean	測定エラー発生時のみ
10	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.9 SW42K0-1000 パルスピックノード

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x4F",
    "unitType": "0x0F",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x4F"
        }
    ],
    "integralPowerConsum ": 45678.9234
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x0F
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	integralPowerConsum	積算電力	Real	0.0 〜 99999999.9999 (Kwh)
7	eepromStatus	EEPROM ステータス	Boolean	EEPROM R/W エラー発生時のみ
8	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.10 SW42DD-1000 電流センサーノード (5A)

■ データサンプル


```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x50",
    "unitType": "0x10",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x50"
        }
    ],
    "distributionSystem12": "1P2W(1CT)",
    "distributionSystem34": "1P2W(1CT)",
    "outputChannel1": true,
    "outputChannel2": true,
    "outputChannel3": false,
    "outputChannel4": false,
    "meanCurrent1": 2.124,
    "meanCurrent2": 3.529,
    "meanCurrent3": 0.0,
    "meanCurrent4": 0.0,
    "instantCurrent1": 2.987,
    "instantCurrent2": 4.201,
    "instantCurrent3": 0.0,
    "instantCurrent4": 0.0,
    "meanPower1": 0.212,
    "meanPower2": 0.352,
    "meanPower3": 0.0,
    "meanPower3": 0.0,
    "instantPower1" 0.298,
    "instantPower2" 0.419,
    "instantPower3" 0.0,
    "instantPower4" 0.0,
    "integralPowerConsum1" 1466789.04,
    "integralPowerConsum2" 2765578.11,
    "integralPowerConsum3" 1.67,
    "integralPowerConsum4" 0.37,
}
```


#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x10
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)

次のページに続く

表 1 – 前のページからの続き

#	JSON キー	内容	データ型	補足
5	routes	経路情報	JSON obj	unitId: 中継ユニットの ID RSSI: 経路情報の RSSI
6	distributionSystem12	CH1,2 の配電方式	String	**1P2W(1CT):** 単相二線式 1CT **3P3W(1CT):** 三相三線式 1CT **1P3W(2CT):** 単相三線式 2CT **3P3W(2CT):** 三相三線式 2CT
7	distributionSystem34	CH3,4 の配電方式	String	前項と同じ
8	outputChannel1	CH1 の出力状態	Boolean	
9	outputChannel2	CH2 の出力状態	Boolean	
10	outputChannel3	CH3 の出力状態	Boolean	
11	outputChannel4	CH4 の出力状態	Boolean	
12	meanCurrent1	CH1 平均電流	Real	0.000 〜 9.999 (A)
13	meanCurrent2	CH2 平均電流	Real	0.000 〜 9.999 (A)
14	meanCurrent3	CH3 平均電流	Real	0.000 〜 9.999 (A)
15	meanCurrent4	CH4 平均電流	Real	0.000 〜 9.999 (A)
16	instantCurrent1	CH1 瞬時電流	Real	0.000 〜 9.999 (A)
17	instantCurrent2	CH2 瞬時電流	Real	0.000 〜 9.999 (A)
18	instantCurrent3	CH3 瞬時電流	Real	0.000 〜 9.999 (A)
19	instantCurrent4	CH1 瞬時電流	Real	0.000 〜 9.999 (A)
20	meanPower1	CH1 平均電力	Real	0.000 〜 9.999 (Kw)
21	meanPower2	CH2 平均電力	Real	0.000 〜 9.999 (Kw)
22	meanPower3	CH3 平均電力	Real	0.000 〜 9.999 (Kw)
23	meanPower4	CH4 平均電力	Real	0.000 〜 9.999 (Kw)
24	instantPower1	CH1 瞬時電力	Real	0.000 〜 9.999 (Kw)
25	instantPower2	CH2 瞬時電力	Real	0.000 〜 9.999 (Kw)
26	instantPower3	CH3 瞬時電力	Real	0.000 〜 9.999 (Kw)
27	instantPower4	CH4 瞬時電力	Real	0.000 〜 9.999 (Kw)
28	integralPowerConsum1	CH1 積算有効電力量	Real	0.00 〜 9999999.99 (Kwh)
29	integralPowerConsum2	CH2 積算有効電力量	Real	0.00 〜 9999999.99 (Kwh)
30	integralPowerConsum3	CH3 積算有効電力量	Real	0.00 〜 9999999.99 (Kwh)
31	integralPowerConsum4	CH4 積算有効電力量	Real	0.00 〜 9999999.99 (Kwh)
32	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.11 SW42DD-1100 電流センサーノード (200A)

■ データサンプル


```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x51",
    "unitType": "0x11",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x51"
        }
    ],
    "distributionSystem12": "3P3W(1CT)",
    "distributionSystem34": "3P3W(1CT)",
    "outputChannel1": true,
    "outputChannel2": true,
    "outputChannel3": false,
    "outputChannel4": false,
    "meanCurrent1": 27.4,
    "meanCurrent2": 31.9,
    "meanCurrent3": 0.0,
    "meanCurrent4": 0.0,
    "instantCurrent1": 32.1,
    "instantCurrent2": 35.8,
    "instantCurrent3": 0.0,
    "instantCurrent4": 0.0,
    "meanPower1": 9.52,
    "meanPower2": 11.08,
    "meanPower3": 0.0,
    "meanPower3": 0.0,
    "instantPower1" 11.15,
    "instantPower2" 12.42,
    "instantPower3" 0.0,
    "instantPower4" 0.0,
    "integralPowerConsum1" 2456789.34,
    "integralPowerConsum2" 3765678.21,
    "integralPowerConsum3" 21.27,
    "integralPowerConsum4" 10.31,
}
```


#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x11
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)

次のページに続く

表 2 – 前のページからの続き

#	JSON キー	内容	データ型	補足
5	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
6	distributionSystem12	CH1,2 の配電方式	String	**1P2W(1CT):** 単相二線式 1CT **3P3W(1CT):** 三相三線式 1CT **1P3W(2CT):** 単相三線式 2CT **3P3W(2CT):** 三相三線式 2CT
7	distributionSystem34	CH3,4 の配電方式	String	前項と同じ
8	outputChannel1	CH1 の出力状態	Boolean	
9	outputChannel2	CH2 の出力状態	Boolean	
10	outputChannel3	CH3 の出力状態	Boolean	
11	outputChannel4	CH4 の出力状態	Boolean	
12	meanCurrent1	CH1 平均電流	Real	0.0 〜 999.9 (A)
13	meanCurrent2	CH2 平均電流	Real	0.0 〜 999.9 (A)
14	meanCurrent3	CH3 平均電流	Real	0.0 〜 999.9 (A)
15	meanCurrent4	CH4 平均電流	Real	0.0 〜 999.9 (A)
16	instantCurrent1	CH1 瞬時電流	Real	0.0 〜 999.9 (A)
17	instantCurrent2	CH2 瞬時電流	Real	0.0 〜 999.9 (A)
18	instantCurrent3	CH3 瞬時電流	Real	0.0 〜 999.9 (A)
19	instantCurrent4	CH1 瞬時電流	Real	0.0 〜 999.9 (A)
20	meanPower1	CH1 平均電力	Real	0.00 〜 99.99 (Kw)
21	meanPower2	CH2 平均電力	Real	0.00 〜 99.99 (Kw)
22	meanPower3	CH3 平均電力	Real	0.00 〜 99.99 (Kw)
23	meanPower4	CH4 平均電力	Real	0.00 〜 99.99 (Kw)
24	instantPower1	CH1 瞬時電力	Real	0.00 〜 99.99 (Kw)
25	instantPower2	CH2 瞬時電力	Real	0.00 〜 99.99 (Kw)
26	instantPower3	CH3 瞬時電力	Real	0.00 〜 99.99 (Kw)
27	instantPower4	CH4 瞬時電力	Real	0.00 〜 99.99 (Kw)
28	integralPowerConsum1	CH1 積算有効電力量	Real	0.00 〜 9999999.99 (Kwh)
29	integralPowerConsum2	CH2 積算有効電力量	Real	0.00 〜 9999999.99 (Kwh)
30	integralPowerConsum3	CH3 積算有効電力量	Real	0.00 〜 9999999.99 (Kwh)
31	integralPowerConsum4	CH4 積算有効電力量	Real	0.00 〜 9999999.99 (Kwh)
32	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.12 SW42D0-1000 電流センサーノード

■ データサンプル


```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x52",
    "unitType": "0x12",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x52"
        }
    ],
    "channel1LowCurrent": false,
    "distributionSystem12": "3P3W(1CT)",
    "distributionSystem34": "3P3W(1CT)",
    "outputChannel1": true,
    "outputChannel2": true,
    "outputChannel3": false,
    "outputChannel4": false,
    "current1": 27.4,
    "current2": 31.9,
    "current3": 0.0,
    "current4": 0.0,
    "power1": 9.52,
    "power2": 11.08,
    "power3": 0.0,
    "power3": 0.0,
    "integralPowerConsum1" 2456789.34,
    "integralPowerConsum2" 3765678.21,
    "integralPowerConsum3" 21.27,
    "integralPowerConsum4" 10.31,
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x12
4	batteryVoltage	電池電圧の状態コード	String	**0x00:** 2.8V 以上 **0x01:** BLD1(2.5〜2.8V) **0x02:** BLD2(2.5V 未満)
5	routes	経路情報	JSON obj	unitId: 中継ユニットの ID RSSI: 経路情報の RSSI
6	distributionSystem12	CH1,2 の配電方式	String	**1P2W(1CT):** 単相二線式 1CT **3P3W(1CT):** 三相三線式 1CT **1P3W(2CT):** 単相三線式 2CT **3P3W(2CT):** 三相三線式 2CT
7	distributionSystem34	CH3,4 の配電方式	String	前項と同じ
8	channel1LowCurrent	CH1 の電流が 1A 未満	Boolean	
9	outputChannel1	CH1 の出力状態	Boolean	
10	outputChannel2	CH2 の出力状態	Boolean	
11	outputChannel3	CH3 の出力状態	Boolean	
12	outputChannel4	CH4 の出力状態	Boolean	
13	current1	CH1 電流	Real	0.0 〜 999.9 (A)
14	current2	CH2 電流	Real	0.0 〜 999.9 (A)
15	current3	CH3 電流	Real	0.0 〜 999.9 (A)
16	current4	CH4 平均電流	Real	0.0 〜 999.9 (A)
17	power1	CH1 電力	Real	0.0 〜 999.9 (Kw)
18	power2	CH2 電力	Real	0.0 〜 999.9 (Kw)
22	power3	CH3 電力	Real	0.0 〜 999.9 (Kw)
23	power4	CH4 平均電力	Real	0.0 〜 999.9 (Kw)
24	integralPowerConsum1	CH1 積算有効電力量	Real	0.0 〜 99999999.9 (Kwh)
25	integralPowerConsum2	CH2 積算有効電力量	Real	0.0 〜 99999999.9 (Kwh)
26	integralPowerConsum3	CH3 積算有効電力量	Real	0.0 〜 99999999.9 (Kwh)
27	integralPowerConsum4	CH4 積算有効電力量	Real	0.0 〜 99999999.9 (Kwh)
32	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.13 SW4230-1000 CO2 ノード

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x60",
    "unitType": "0x20",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x60"
        }
    ],
    "co2": 407
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x20
4	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
5	co2	二酸化炭素濃度	Integer	0 〜 99999 (ppm)
6	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.14 SW4260-1x10 電力モニターノード

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x61",
    "unitType": "0x21",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x61"
        }
    ],
    "integralPowerConsum": 723619.23
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x21
4	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
5	integralPowerConsum	積算電力量	Real	0.00 ～ 999999999.00 (Kwh)
6	sensorConnected	接続センサー番号	String	電源投入時のみ. 0xNNNNNNNN
7	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.15 SW4280-1000 Modbus RTU ノード

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x63",
    "unitType": "0x23",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x63"
        }
    ],
    "length": 10,
    "modbusId": "0x01",
    "unitId": "0x63",
    "modbusFunction":"0x12FAB101C1FEA010"
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x23
4	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
5	length	Modbus メッセージ長	Integer	3 〜 10. *1
6	modbusId	Modbus のユニット ID	String	0xNN. *1
7	unitId	SID データの送信元 ID	String	*1
8	modbusFunction	Modbus ファンクション番号	String	0xNN. *1
9	modbusData	取得したデータ	String	0xNNNNNNNNNNNNNNNN. *1
7	swVersion	ソフトウェアバージョン	String	電源投入時のみ

注釈:

1. 下流方向制御等によって SW-4280-1000 Modbus RTU ノードより、polls オブジェクトに設定されていないメッセージが、送られて来た場合にのみ適用されます.

polls オブジェクトに設定されたポーリング処理の出力は、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。

63.16 SW4260-1x20 電力モニターノード

■ データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0x68",
    "unitType": "0x28",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0x68"
        }
    ],
    "integralPowerConsum": 723619.23
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x28
4	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
5	integralPowerConsum	積算電力量	Real	0.00 〜 999999999.00 (kwh)
6	sensorConnected	接続回路番号	String	電源投入時のみ. 0xNNNNNNNN
7	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.17 SW4100-1000 ルーターノード

■ データサンプル

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

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0xEF
4	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
5	periodic	8 時間毎に送信されるフラグ	Boolean	正常に機能していることを示す.
6	swVersion	ソフトウェアバージョン	String	電源投入時のみ

63.18 ベースノード

■ データサンプル

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

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	**0xFA:** SB4A00-1000 Armadillo-IoT G3L 用アドオンモジュール **0xFB:** SB4020-1000 OpenBlocks シリーズ アドオンモジュール **0xFC:** SW4000-2000 ネットワークスパイユニット **0xFD:** SW4000-1000 USB ベース **0xFE:** SW4300-1000 Ethernet ベース **0xFF:** SW4500-1000 RS-232C ベース
4	routes	経路情報	JSON obj	**unitId:** 中継ユニットの ID **RSSI:** 経路情報の RSSI
5	periodic	8 時間毎に送信されるフラグ	Boolean	正常に機能していることを示す.
6	swVersion	ソフトウェアバージョン	String	電源投入時のみ