データサブシステム
API解説

エブリセンスジャパン株式会社

0.9版 2015年 12月 16日

データサブシステム解説

本ドキュメントは、データサブシステムについて解説します。

以下の章から成っています。

データサブシステムの概念と構造

データサブシステムとは

EverySenseシステムは、「サーバ」と「データサブシステム」から構成されています。

サーバでは、

- レシピの作成
- ファームの管理
- 会計処理

といった、センサーデータをやりとりするための、契約等の処理を主に行います。

データサブシステムは、サーバによって作られた契約情報を元に、

- データの振り分け処理
- 会計基礎データの作成
- デバイスやデータシンクとのデータ通信

といった、実際のデータのやりとりと、それによって発生する課金等の処理を行います。

データサブシステムから見たEverySenseシステムの構成を、以下に示します。

ファームオーナの所有するデバイスからの情報は、「デバイスゲートウェイ」を経由して「メッセージルータ」に入ります。この「メッセージルータ」がデバイスからのデータをレシピに結びつけて配送します。配送されたデータは「出力ゲートウェイ」を経由して、レストランオーナの持っているデータシンクが受け取ります。

デバイスは様々なプロトコルでデータを送信しますが、それらのプロトコルからデータサブシステム内のプロトコルへの変換は、「デバイスゲートウェイ」が行います。現在のところ用意されているデバイスゲートウェイは、JSON over HTTP(s)に対するゲートウェイです。この他のプロトコルへの対応は、標準的なプロトコル(たとえばMQTT等)への対応は、随時EverySense社の方で開発します。それ以外の特殊なプロトコルに関しては、デバイスベンダー様が開発の上の持ち込みとなります。

API説明書(内部API)

ここではデータサブシステムが提供するAPIについて説明します。このAPIはデータサブシステムがデバイスゲートウェイに提供しているインターフェイスなので、実際にデバイスゲートウェイが外部に公開するAPIは、ここで挙げられているAPIとは異なる可能性があります。

概要

デバイスとEverysense Serverとの通信は、デバイスゲートウェイを通じて行われます。Everysense Serverがデバイスと正しく通信するためには、デバイスゲートウェイは以下のことが出来る必要があります。

1. デバイスより受信したデータをデータサブシステムに送る
2. 受信したセンサーデータの単位系の整合
3. (必要なら)デバイスの認証
4. Everysense Serverの必要に応じて、デバイスへのpush

本ドキュメントでは、これらについてAPIの解説を行います。

前提事項

デバイスゲートウェイに提供するAPIの基本

デバイスゲートウェイトEverySense Serverとの間の通信は、全てMessagePack RPC を使います。MessagePack および MessagePack RPC についての詳しい説明については、当該ページを御覧下さい。仕様については、MessagePack specification にあります。

デバイスゲートウェイに提供するAPIのデータの型については、以下のものを使用します。

型名	意味
Integer	整数
Nil	nil
Boolean	論理値、true or false
Float	浮動小数点数
String	UTF-8文字列
Array	順序のあるオブジェクトの並び
Map	フィールド名のついたオブジェクトの並び

時刻は文字列に変換して扱われます。

APIは論理的には、

```
API名(引数1, 引数2,...)
```

のように呼び出されます(実際にどう表現されるかは言語に依ります)。このAPI解説では

呼称	型	意味
引数1	引数1の型	引数1の意味
引数2	引数2の型	引数2の意味

のような形式で説明されています。

型が*Map*の場合、値には複数のフィールドが名前と共に列挙されます。この解説では、

名前	意味	型
name	このフィールドの意味	このフィールドの型

のように表現されます。

提供するAPIの種類

APIは以下のようなものがあります。

- セションAPI
- メッセージAPI
- オーダーAPI
- デバイスAPI
- センサーAPI
- ファームAPI
- ファームオーナAPI
- オーナAPI
- データ出力API

これらのうち、センサーデバイスで主に使うものは、

- メッセージAPI

です。

また、レストランオーナとしてデータを取得するのに使うものは、

- データ出力API

です。

これ以外のAPIは主にはEveryPostのような、Every Sense serverを操作する機能を持ったデバイスのためのAPIです。

APIの呼出方法

APIはMessagePackRPCか、JSON over HTTPによって呼び出すことが出来ます。

いずれの呼び出しも、プリミティブなライブラリを組み合わせて呼び出すことが出来ますが、Rubyからの呼び出しについてはOpenESが用意されています。以下はその例です。

MessagePackRPC

MessagePackRPCでAPIを呼び出す場合は、'every_sense_msgpack_client.rb'を使います。

たとえば、'put_message'を呼び出す場合は、

```
require 'every_sense_msgpack_client'
server = EverySense::MsgPackClient.new('api.every-sense.com')
p server.put_message("fc1bfbd8-2b20-473e-8cd6-6030b2402ec1",
                     [
                      {
                        "data" => {
                          "at" => Time.now.to_s,
                          "unit" => "degree",
                          "location" => ["35.705570","139.770767"],
                          "elevation" => "0.000000",
                          "datum" => "WGS84",
                          "memo" => ""
                        },
                        "sensor_name" => "GPS"
                      }])
```

のように行います。

JSON

APIをJSONから呼ぶ場合は、

http://<ホスト名>:<ポート番号>/<メソッド名>

の形式のURLにPOSTします(一部の参照系のAPIではGETもあります)。

- ホスト名
  現在は'api.every-sense.com'です
- ポート番号
  現在は7001です。8001を使うとhttpsでアクセスされます。
- メソッド名
  APIのメソッド名そのものです
- POSTの内容
  JSONの配列であり、要素の順序はAPIの引数の順序に一致させます

たとえば、'resolv'を呼び出す場合は、

http://<ホスト名>:<ポート番号>/resolv

に

[<デバイスUUID>，<ローカル名>]

をPOSTします。

Rubyから呼び出す場合は、'every_sense_json_client.rb'を使います。

たとえば、'put_message'を呼び出す場合は、

```
require 'every_sense_json_client'
server = EverySense::JsonClient.new
p server.put_message("fc1bfbd8-2b20-473e-8cd6-
6030b2402ec1",
                     [
                      {
```

```
                          "data" => {
                            "at" => Time.now.to_s,
                            "unit" => "degree",
                            "location" =>
["35.705570","139.770767"],
                            "elevation" => "0.000000",
                            "datum" => "WGS84",
                            "memo" => ""
                          },
                          "sensor_name" => "GPS"
                        }])
```

のように行います。

APIの認証

プライベートなデータの参照およびデータの更新を行うAPIには、認証を行うものがあります。

返り値

多くのAPIでは、以下のフィールドを含む*Map*形式で結果が返ります。これ以外の結果を返すものもあります。

特にことわりのない場合、APIの説明は以下のフィールドについての説明を省略します。

名前	型	意味
code	Integer	復帰コード
reason	Text	エラー説明(復帰コードが-1, -2の場合)
message	Text	エラ-メッセージ(復帰コードが-10, -20の場合)
trace	Text	エラートレース(復帰コードが-20の場合)

復帰コード

- 0: 成功
- -1: 失敗あるいはデータなし
- -2: 認証エラー
- -10:デバイスゲートウェイ内でのエラー(通常パラメータ指定の間違い)
- -20: データサブシステムのバグによるエラー

セションAPI

概要

セションAPIは、認証済みのセションを管理するAPIです。

セションは認証によって作成され、作成後一定期間(現在は1週間)有効です。

セションが使用されると、使用された時から一定期間(現在は1週間)、有効期間が延長されます。

セションの正当性は、セションキーの検証によってのみ行われます。セションキーが第三者に漏洩した場合は、すみやかに無効化する必要があります。

現在のところ、セションでの認証が有効なAPIは、データ出力APIのみです。

API

セションについてのAPIは、以下のものがあります。

- create_session
  認証用のセションを作ります
- check_session
  セションが有効かどうか調べます
- delete_session
  セションを無効にします

create_session

概要

セションを作成します

説明

create_sessionはセションを作成し、セションキーを返します。

String ユーザーUUID

String パスワード

String セションの説明(省略可)

返り値

Map 返り値

Integer 復帰コード(code)

0: 成功

-2: 認証エラー

String セションキー(session_key)

成功しなかった場合はsession_keyは返りません。

Edit

check_session

概要

セションが有効かどうか調べます。

説明

check_sessionは、与えられたセションキーが有効かどうか調べます。

String セションキー

Map 返り値

Integer 復帰コード(code)

0: 成功

-1: セションなし

Edit

delete_session

概要

セションを削除します

説明

delete_sessionは与えられたセションキーを無効化します。

String ユーザーUUID

String パスワード

String セションキー

返り値

Map 返り値

Integer 復帰コード(code)

0: 成功

-1: セションなし

-2: 認証エラー

メッセージAPI

概要

ファームで収集したセンサーデータを、レシピに従いレストランに配送します。

デバイスゲートウェイから見た場合、デバイスゲートウェイが受けつけたデータを吐き出す先になります。

API

デバイスゲートウェイからメッセージルータに対するAPIの機能(function)は、以下のものがあります。

- put_message
  受信したデータをメッセージルータに送ります。
- resolv
  センサーローカル名からセンサーUUIDを求めます。
- get_message
  入力キューにある受信データを取得します。

put_message

概要

受信したデータを、メッセージルータに送ります。

説明

put_messageは、デバイスゲートウェイからのデータを、メッセージルータに送ります。

呼称	型	意味
デバイスUUID	String	デバイスに付与されたuuid

データの並び	Array of Object	ファームから収集されたデータ

個々のセンサーデータは、以下のようになります

名前	呼称	型
sensor_uuid	センサーUUID	String
sensor_name	センサーローカル名	String
data	センサーデータ	Object

「センサーローカル名」または「センサーUUID」で識別されます、「センサーローカル名」と「センサーUUID」はいずれかを指定します。

センサーデータの具体的な内容については、センサーデータにあります。

返り値

標準の返り値が返ります。

デバイスUUIDが正当な場合は、見掛け上処理は成功します。ただし、実際にデータが正しく配送されるかどうかは、データの内容やレシピの状態に依存します。

resolv

概要

センサーローカル名からセンサーUUIDを求めます。

説明

resolvは、デバイスとセンサー名からセンサーUUIDを求めます。

呼称	型	意味
デバイスUUID	String	デバイスに付与されたuuid
センサーローカル名	String	デバイス上のセンサーにつけられた名前

返り値

返り値は以下のフィールドを持つMAPです。この他に標準のフィールドを持ちます。

名前	型	意味
uuid	String	センサーUUID

get_message

概要

入力キューにある受信データを取得します。

説明

get_messageは、メッセージルータの入力キューにある受信データを取得します。

呼称	型	意味
ユーザuuid	String	ユーザのUUID
パスワード	String	
デバイスUUID	String	データを取得するデバイスのUUID

返り値

以下のフィールドを持つ*Map*です。この他に標準のフィールドを持ちます。

名前	型	意味
count	Integer	データ件数
inputs	Object Array	データ

出力されるデータの形式については、出力されるデータの形式を参照して下さい。

データ出力API

概要

レストランに配送されたデータを外部に取り出すためのものです。

API

データサブシステムからデータを取り出すためのAPIは、以下のものがあります。

- get_output_data
  データを取り出します
- clear_output_data
  出力バッファのデータを削除します

get_output_data

概要

出力バッファにあるデータを取り出します。

説明

get_putput_dataは、出力バッファにあるデータを取り出します。

名前	型	意味
user_uuid	String	ユーザUUID
login_name	String	ログイン名
password	String	パスワード
session_key	String	セションキー
recipe_uuid	String	レシピUUID
keep	String	データを保持するかどうか('true' or 'false')

limit	Integer	最大取り出し件数 1度のアクセスで取り出す最大件数を指定します。デフォルトは1000です
format	String	出力フォーマット データ出力の形式を指定します。デフォルトは'JSON'です。 XMLも指定可能です
from	String	出力開始時刻 デフォルトは出力バッファの先頭からです。
to	String	出力終了時刻 デフォルトは出力バッファの末尾までです。

このファンクションでは、以下の3とおりの認証方法が提供されています。

- ユーザUUIDとパスワードによる方法
- ログイン名とパスワードによる方法
- セションキーによる方法

呼び出し時に必要なものを指定することにより、認証方法が選択されます。

返り値

成功した場合は、データが指定したフォーマットで返ります。失敗した場合は、標準の返り値で通知されます。

名前	型	意味
code	Integer	復帰コード
result	Text	出力データ

clear_output_data

概要

出力ファッバにあるデータを削除します。

説明

clear_output_dataは、出力バッファにあるデータを削除します。

get_output_dataでkeepを指定した場合の後処理に使うことを想定しています。

呼称	型	意味
ユーザUUID	String	
パスワード	String	
レシピUUID	String	ダウンロードするデータのレシピを指定する
オプション	Map	ダウンロードのオプション。詳細後述

オプション

名前(タグ)	意味	型
from	出力開始時刻 デフォルトは出力バッファの先頭からです。	String
to	出力終了時刻 デフォルトは出力バッファの末尾までです。	String

返り値

標準の返り値です。

JSON over HTTPゲートウェイとのやりとり

現在のデータサブシステムは、HTTPでJSONをやりとりするゲートウェイが用意されています。基本的には既に延べたAPIをJSONとHTTPを使うというように読み換えることで使えますが、よく使われるAPIだと思われるので、あらためて解説します。

パラメータの意味等の詳細については、API説明書を参照して下さい。

アクセスするURL

EverySenseのJSON over HTTPゲートウェイは、以下のURLです。

```
http://api.every-sense.com:7001/
```

また、HTTPSの場合は、

```
https://api.every-sense.com:8001/
```

です。

用意されているAPI

以下のAPIが用意されています。

機能名	メソッド	意味
device_data	POST	デバイスからデータを送ります。内部的にはput_messageです。
device_data	GET	デバイスから送られたデータを読み出します。このAPIは主にデバイスデータを確認するために使われます。
recipe_data	GET	レシピに届いたデータを読み出します。内部的には、get_output_dataです。
session	POST	認証を行い、セションを作成し、セションキーを取得します
session	DELETE	セションを無効にします

これら以外のAPIについては、API説明書にあるAPIと同じ名前で`POST`することで利用可能ですが、現在整理中です。

APIと引数

POST device_data

呼び出し方

```
/device_data/<デバイスUUID>
```

データ

センサーデータをJSONで表現したものをデータとして渡します。

返り値

正しく処理された場合は、put_messageの返り値、すなわち成功した場合の標準の返り値がJSON形式で返ります。具体的には、以下のようなJSONです。

```
{
    code: "0"
}
```

ゲートウェイ内でエラーが発生した場合は、`'code'`が`-10`、`'message'`が「エラー理由」のJSONが返ります。それ以外のエラーも'code'が負数となって、エラーメッセージ等が返ります。一般的には、以下のような形式のJSONです。

```
{
    code: "<復帰コード>",
    reason: "<エラー発生の理由>",
    message: "<エラーメッセージ>",
    trace: "<サーバプロセスのバックトレース>"
}
```

GET device_data

呼び出し方

```
/device_data/<デバイスUUID>?<オプション>
```

オプション

以下のものが指定可能です。

オプション	意味	説明
user_uuid	ユーザーUUID	認証情報としてユーザを識別するのに使います
login_name	ログイン名	認証情報としてユーザを識別するのに使います。 user_uuidと同時に指定した場合、user_uuidを優先します
password	パスワード	認証のためのパスワードです。user_uuidまたはlogin_nameを指定した場合に使います。 平文の上、URLで見えてしまうので、注意して使って下さい
session_key	セションキー	POST sessionで作成したセションキーを指定します キーの無効化も簡単なので、パスワードを使った認証よりも安全です
limit	取得上限件数	一度に取得するデータ件数の上限を指定します デフォルトは1000なので、それ以上取得したい場合は、明示します。
from	取得開始時刻	取得するデータの開始時刻です
to	取得終了時刻	取得するデータの終了時刻です

返り値

正しく処理された場合は、put_messageの返り値、すなわち成功した場合の標準の返り値がJSON形式で返ります。具体的には、以下のようなJSONです。

```
{
    code: "0"
}
```

ゲートウェイ内でエラーが発生した場合は、'code'が-10、'message'が「エラー理由」のJSONが返ります。それ以外のエラーも'code'が負数となって、エラーメッセージ等が返ります。一般的には、以下のような形式のJSONです。

```
{
    code: "<復帰コード>",
    reason: "<エラー発生の理由>",
    message: "<エラーメッセージ>",
    trace: "<サーバプロセスのバックトレース>"
}
```

ただし、エラーの種類のよっては、全てがない場合もあります。

GET recipe_data

呼び出し方

```
/recipe_data/<レシピUUID>.<フォーマット>?<オプション>
```

オプション

以下のものが指定可能です。

オプション	意味	説明
user_uuid	ユーザーUUID	認証情報としてユーザを識別するのに使います
login_name	ログイン名	認証情報としてユーザを識別するのに使います。 user_uuidと同時に指定した場合、user_uuidを優先します
password	パスワード	認証のためのパスワードです。user_uuidまたはlogin_nameを指定した場合に使います。 平文の上、URLで見えてしまうので、注意して使って下さい
session_key	セションキー	POST sessionで作成したセションキーを指定します キーの無効化も簡単なので、パスワードを使った認証よりも安全です
limit	取得上限件数	一度に取得するデータ件数の上限を指定します デフォルトは1000なので、それ以上取得したい場合は、明示します。

from	取得開始時刻	取得するデータの開始時刻です
to	取得終了時刻	取得するデータの終了時刻です
keep	データの保管	'true'を指定すると、データ取得後にデータ削除を行いません。
inline	インライン指定	'true'を指定すると、ファーム情報をインラインでデータに埋め込みます。 当然、公開されているもののみです。
format	フォーマット	正常に処理された時に返すデータのフォーマットを指定します。 通常、フォーマットはURLのパス部で指定されますが、オプションで変更することも出来ます。 現在指定可能なものは、XML, JSONです。

返り値

正しく処理された場合は、HTTPステータスが200で、指定したフォーマットでレシピに収集されたデータが出力されます。

何らかのエラーが発生した場合、HTTPステータスが500で、JSONで以下の形式のメッセージが出力されます。

```
{
    code: "-10",
    message: "<エラーメッセージ>",
}
```

POST session

呼び出し方

```
/session
```

データ

以下のデータをJSON形式で渡します。

オプション	意味	説明
user_uuid	ユーザーUUID	認証情報としてユーザを識別するのに使います
login_name	ログイン名	認証情報としてユーザを識別するのに使います。 user_uuidと同時に指定した場合、user_uuidを優先します
password	パスワード	認証のためのパスワードです。user_uuidまたはlogin_nameを指定した場合に使います。 平文の上、URLで見えてしまうので、注意して使って下さい

返り値

正しく認証された場合は以下のようなJSONが返ると共に、セションが生成されます。このセションキーを使って、認証が必要なAPIをアクセスします。

```
{
    code: "0",
    session_key: "<セションキー>",
}
```

エラーのあった場合は、以下のようなJSONが返ります。

```
{
    code: "<復帰コード>",
    message: "<メッセージ>",
}
```

DELETE session

呼び出し方

```
/session/<セションキー>
```

返り値

与えたセションキーに対応するセションが存在した場合は、

```
{
    code: 0
}
```

が返り、セションキーは無効になります。

与えたセションキーに対応するセションが存在しない場合は、

```
{
    code: "-1",
    reason: "session not found"
}
```

が返ります。

センサーデータ

センサーデータの種類

基本的なセンサーについて、そのデータ形式を規定します。

- 気温センサーデータ
- 水温センサーデータ
- 湿度センサーデータ
- 土壌温度センサーデータ
- 電気伝導度センサーデータ
- 降水量センサーデータ
- 気圧センサーデータ
- 風向センサーデータ
- 風速センサーデータ
- 近接センサーデータ
- 加速度センサーデータ
- ジャイロセンサーデータ
- 方位センサーデータ
- 磁気センサーデータ
- 騒音センサーデータ
- 歩数センサーデータ
- モーションアクティビティセンサーデータ
- 位置センサー

センサーデータを表現する型について

特定のプログラム言語等に依存しないでデータ形式を表現するために、以下の抽象型を定義します。

- Number

数値(スカラー)を表現します
- GeoLocation
  地理的位置を表現します
- Dimension
  寸法や距離といった、物理的な大きさを表現します
- TimeStamp
  日付を含む時刻を表現します
- Text
  文書を表現する文字列で、改行等を含みます
- String
  文字列で、改行等を含みません
- Symbol
  名前です
- Enumerate
  1つ以上のSymbolからなる集合のSymbolを値とします

共通フィールドについて

以下のフィールドは、センサー共通のフィールドです。ただし、センサーの性質によっては存在しないこともあります。

名前	意味	型
location	センサー設置場所	Location
unit	単位	String
accuracy	精度	Number
memo	覚書	Text
at	値取得時刻	TimeStamp

精度は相対精度として、標準単位は"percent"です。

時刻はタイムゾーン情報を含めます。ない場合はUTCとみなします。

気温センサーデータ

名前	意味	型
value	センサー値	Number

標準の単位は"degree_Celsius"です。

水温センサーデータ

名前	意味	型
depth	センター設置水深	Dimension
value	センサー値	Number
depth_unit	センサー設置水深の単位	String

標準の単位は"degree_Celsius"、水深の単位は"m"です。

湿度センサーデータ

名前	意味	型
value	センサー値	Number

標準の単位は"%RH"です。

土壌温度（soil temperature）センサーデータ

名前	意味	型
depth	センサー設置深さ	Dimension
value	センサー値	Number
depth_unit	センサー設置深さの単位	String

標準の単位は"degree Celsius"です。センサー設置深さの単位は"cm"です。

電気伝導度 EC センサーデータ

名前	意味	型
depth	センサー設置深さ	Dimension
value	センサー値	Number
depth_unit	センサー設置深さの単位	String

標準の単位は"mS/mm"、設置深さの単位は"cm"

降水量

名前	意味	型
value	センサー値	Number

標準の単位は"mm"。

気圧センサーデータ

名前	意味	型
value	センサー値	Number

標準の単位は"hPa"

風向センサーデータ

名前	意味	型
value	センサー値	Number

標準単位は"degree"

風速センサーデータ

名前	意味	型

value	センサー値	Number

標準の単位は"m/s"

近接センサーデータ

名前	意味	型
value	センサー値	String("on" or "off")

単位を持ちません。

加速度センサーデータ

名前	意味	型
values	[センサー値X,センサー値Y,センサー値Z]	Number Array

標準単位は"m/s^2"です。

iPhone等では"G"で取得されると思いますが、変換するかunitに"G"を指定します。

ジャイロセンサーデータ

名前	意味	型
values	[センサー値roll,センサー値pitch,センサー値yaw]	Number Array

標準単位は"deg/s"です。

方位センサーデータ

名前	意味	型
value	方位	Number
unit	方位単位	String
north	北の種類	String("magnetic" or "true")

方位の標準単位は"degree"です。北の種類の標準は"true"です。

磁気センサーデータ

名前	意味	型
values	[センサー値X,センサー値Y,センサー値Z]	Number Array

磁気センサー値の標準単位は"nT"。

騒音センサーデータ

名前	意味	型
value	センサー値	Number

標準単位は"dB"です。

歩数センサーデータ

名前	意味	型
value	センサー値	Number

単位は"step"のみです。

モーションアクティビティセンサーデータ

名前	意味	型
value	センサー値	String

単位はありません。

位置センサー

名前	意味	型
values	[経度,緯度,標高]	Dimension

datum	測地系	String

標準の単位は"degree", 測地系は"WGS84"です。

出力されるデータの形式

既に説明したように、データサブシステムから出力可能なデータは、

- レシピに振り分けされたセンサーデータ
- デバイスから送信されたセンサーデータ

とがります。基本的に、センサーデータの内容は同じでですが、前者は振り分けの際に、

- ファームで公開されているデータ
- デバイスを識別するためのUUID
- センサーのクラス名(センサーの種類の名前)
- センサーデータのクラス名(データの単位の名前)

等が付加されています。

実際に出力されるデータは、このデータをそれぞれのデータフォーマット(JSON, XML等)に合わせてserializeされたものです。

以下にその詳細を示します。

振り分けされたデータの形式(get_output_dataで取得されるデータ)

振り分けされたデータは、以下に示す形式のエントリの配列を1つのデバイスのデータレコードとします。

名前	意味	説明	型
device_uuid	デバイスを表現するUUID	このUUIDが同じデータは、同じデバイスから出力されていることが保証されています。レシピが異なる場合は、同じデバイスでも別のUUIDとなります。	String(36)
sensor_name	センサー	レシピで設定した仮想的なセンサー名	String

	名		
sensor_class_name	センサークラス名	センサーを表現するクラス名、詳細はセンサーリポジトリを参照のこと	String
data_class_name	センサーデータクラス名	測定値を表現するクラス名、たとえば"Accelation(加速度)"といったもの。詳細はセンサーデータクラス名の解説を参照のこと	String
data	センサーデータ	センサーのデータ。形式についてはセンサーデータの解説を参照のこと	Map

`sensor_name`は、レシピ作成の時に指定します。

レシピ従って収集されるデータは、「`sensor_name`を持った仮想的なセンサーを持ったデバイスからのデータ」という考え方をします。具体的にファームのどのセンサーがマッピングされるかについては、レシピの要求とファームの内容とのマッチングによって決定されます。

デバイスからの送信データ(get_messageで取得されるデータ)

デバイスから送信されたデータは、以下に示す形式のエントリの配列を1回のセンスのデータレコードとします。

名前	意味	説明	型
sensor_name	センサー名	デバイスローカルなセンサー名	String
data	センサーデータ	センサーのデータ。形式についてはセンサーデータの解説を参照のこと	Map

センサーのリポジトリ

センサーの名寄せを行うためのリポジトリについて記述したもので、以下のことについて記述しています。

- 基本的な考え方
- 分類
- 対象

基本的な考え方

- センサークラスの呼び方は、[<parameter>] <measurement> of <target> [<modifier>] の形式である
- センサークラスの名前は、「呼び方」を元に構成される
- リポジトリにないmeasurement, parameter, target, modifierは、随時追加される
- センサー値は1つ以上のフィールドを持ち、同じセンサークラスのセンサは同じフィールドを持つ

センサークラスの例

呼び方	クラス名	意味
temperature of water	WaterTemperature	水温
PM 2.5 number of air	AirPM25Number	空気中のPM2.5の数
location of person	PersonLocation	人の位置

measurementの語彙

名前	シンボル
温度	temperature
湿度	humidity
圧力	pressure

変位	displacement
角度	angular
位置	location
照度	illuminance
速度	velocity
加速度	acceleration
頻度	frequency
放射線	radiation
濃度	concentration
数	number

targetの語彙

名前	シンボル
空気	air
水	water
方向	direction
土壌	soil
歩	step
人	person