Guidebook of Machine Readable Appendix (MRA)

改定履歴

日付	版	説明
2021.12.01	1.0.0	1st release
2022.05.27	1.1.0	2nd release number_0-100%をnumber_0-100percentへ変更

・エコーネットコンソーシアムが発行している規格類は、工業所有権(特許,実用新案など)に関する抵触の有無に関係なく制定されています。 エコーネットコンソーシアムは、この規格類の内容に関する工業所有権に対して、一切の責任を負いません。
・この書面の使用による、いかなる損害も責任を負うものではありません。

1. Abstract

APPENDIX ECHONET 機器オブジェクト詳細規定（以下、APPENDIXと略す）の内容をプログラムで容易に利用できるようにするため、APPENDIX に記述された情報を JSON format のデータファイルとして提供する。これを Machine Readable Appendix (MRA) と呼ぶ。このドキュメントは MRA のデータフォーマットを説明する。

APPENDIX には、ECHONET Lite プロトコルのプロパティだけではなく、ECHONET プロトコルのプロパティ（アクセスルールに GetM や SetM の記述があるプロパティ）も記載されているが、MRA では ECHONET Lite プロトコルのプロパティのみを扱う。

APPENDIX は Release A から始まり、以降 B, C,... と機器やプロパティの追加修正を行ってきた。ECHONET Lite を利用するアプリケーションやサービス開発では、最新の Release に対応した機器だけでなく、既に販売されユーザーのもとにある（以前の Release に対応した）機器のサポートも必須である。また APPENDIX は日本語版と英語版が存在する。これらの事情を考慮し、MRA はデータフォーマットとして 日本語と英語を扱うだけでなく、複数の Release version を扱うことができる。

2. Version number

MRA のバージョンとして、フォーマットバージョンと データバージョンを定義する。後述する meta data に記述される。 フォーマットバージョンは、本ドキュメントのバージョンに同期する。データバージョンは公開するデータを区別するためのバージョンである。 バージョン番号は以下のように記述する。

データフォーマットのバージョン

> xxx.yyy.zzz
> xxx: 互換性のないフォーマットの変更
> yyy: 互換性のあるフォーマットの変更
> zzz: Typo の修正など

データのバージョン

> xxx.yyy.zzz
> xxx: 互換性のないデータフォーマットの変更に対応
> yyy: 機器の追加、APPENDIX Release の更新に対応
> zzz: Typo の修正など

3. Data format of MRA

3.1 全体構成

MRA は以下の構成の複数の JSON file として提供する。

```
mraData
 ├──── metaData.json
 ├──── definitions
 |     └──── definitions.json
```

```
├──── nodeProfile
│     └──── 0x0EF0.json
├──── superClass
│     └──── 0x0000.json
└──── devices
      ├──── 0x0130.json
      ├──── 0x0290.json
      ├──── ...
```

metaData.json はバージョン番号やコピーライトなど MRA 全体の情報を記述する。

definitions.json は、データ表現（schema）のなかで繰り返し使われる記述を短い表現で置き換えるための定義（key：テンプレート名 と value：実際の記述 のペア）を記述したものである。

devices フォルダ内の各 JSON ファイルは、それぞれの機器オブジェクトを記述する。ファイル名は EOJ の上位２バイトを16進数の string で表記したもの（例："0x0130.json"）である。同一 EOJ で APPENDIX の Release version によってクラス名に変更がある場合は、それぞれ別々のファイルで記述し、EOJにサフィックスを付加したファイル名とする（例：0x0288_1.json, 0x0288_2.json）。最新の APPENDIX に対応した内容は、サフィックスの無いファイル名（例：0x0288.json）とする。

0x0000.json は、機器オブジェクトのスーパークラスを記述する。

0x0EF0.json は、Node profile object とそのスーパークラスを記述する。理由は、Node profile のスーパークラスは機器オブジェクトのスーパークラスとは異なることと、これを利用するのは Node profile object だけだからである。

devices 内の JSON file および 0x0EF0.json, 0x0000.json は、3.3 章 device description object のデータフォーマットで記述する。

3.2 meta data

Format

```
{
  "date": <release date>,
  "release": <ECHONET Lite release version>,
  "formatVersion": <data format version number>,
  "dataVersion": <MRA data version number>,
  "note": <any additional information>,
  "copyright": <copyright information>
}
```

Keyword	Type	Required	Description
date	string	yes	release date
release	string	yes	MRA データが参照するAppendix version。例: "M"

Keyword	Type	Required	Description
formatVersion	string	yes	フォーマットバージョン。例: 1.0.2
dataVersion	string	yes	データバージョン。例: 1.1.3
note	object	no	追加情報
note.jp	string	no	追加情報（日本語）
note.en	string	no	追加情報（英語）
copyright	string	yes	copyright

Example

```
{
  "date": "2021-12-01",
  "release": "M",
  "formatVersion": "1.0.0",
  "dataVersion": "1.0.0",
  "note": {
    "ja": "Machine Readable Appendix 公式リリース",
    "en": "Machine Readable Appendix official release"
  },
  "Copyright": "(C) 2021 Kanagawa Institute of Technology, ECHONET CONSORTIUM"
}
```

3.3 device description object

devices 内の JSON file および 0x0EF0.json, 0x0000.json は、device description object で記述する。

Format

```
{
  "eoj": <ECHONET object code>,
  "validRelease": {
    "from": <Release version>,
    "to": <Release version>
  },
  "className": {
    "ja": <class name in Japanese>,
    "en": <class name in English>
  },
  "shortName": <short name of the class name>,
  "elProperties": [
    <property description object>,
    <property description object>,
```

```
    ...
  ]
}
```

Keyword	Type	Required	Description	Example
eoj	string	yes	EOJコードの上位２バイトを16進数表記したもの	0x0130
validRelease	object	yes	機器オブジェクト定義が有効なAppendixのバージョンの範囲	
validRelease.from	string	yes	有効範囲の始まりのAppendixバージョン	A, D
validRelease.to	string	yes	有効範囲の終わりのAppendixバージョン	L, latest (*1)
className	object	yes	クラス名	
className.ja	string	yes	クラス名：日本語	家庭用エアコン
className.en	string	yes	クラス名：英語	Home air conditioner
shortName	string	yes	クラス名のショートネーム (*2)	homeAirConditioner
elProperties	array	yes	property description object を要素とした配列 (*3)	

*1 latest は metaData.release の値を意味する。
*2 "shortName" は ECHONET Lite Web API の Device Description で利用する値である。
*3 Super class に記述のあるプロパティを機器オブジェクトでオーバーライドする場合は、ここで記述する。

Example

```
{
  "eoj":"0x0130",
  "validRelease": {
    "from": "A",
    "to": "latest"
  },
  "className": {
    "ja": "家庭用エアコン",
    "en": "Home Air Conditioner"
  },
  "shortName": "homeAirConditioner",
  "elProperties": [
```

```
      <property description object for EPC:0xA0>,
      <property description object for EPC:0xA1>,
      ...
    ]
  }
```

同一 EOJ に対する複数の定義に関して

同一 EOJ で APPENDIX の Release version によってクラス名に変更がある場合は、それぞれ別々のファイルで記述し、EOJにサフィックスを付加したファイル名とする（例：0x0288_1.json, 0x0288_2.json）。最新の APPENDIX に対応した内容は、サフィックスの無いファイル名（例：0x0288.json）とする。

Example

0x0288_1.json

```
{
  "eoj":"0x0288",
  "validRelease": {
    "from": "A",
    "to": "E"
  },
  "className": {
    "ja": "スマート電力量メータ", ...
  },
  ...
}
```

0x0288.json

```
{
  "eoj":"0x0288",
  "validRelease": {
    "from": "F",
    "to": "latest"
  },
  "className": {
    "ja": "低圧スマート電力量メータ", ...
  },
  ...
}
```

3.4 property description object

エコーネットプロパティは、property description object を利用して記述する。

Format

```
{
  "epc": <EPC>,
  "validRelease": {
    "from": <Release version>,
    "to": <Release version>
  },
  "propertyName": {
    "ja": <property name in Japanese>,
    "en": <property name in English>
  },
  "shortName": <short name of the property name>,
  "accessRule": {
    "get": <GET access rule>,
    "set": <SET access rule>,
    "inf": <Anno access rule>
  },
  "descriptions": {
    "ja": <description in Japanese>,
    "en": <description in English>
  },
  "atomic": <EPC of atomic operation>,
  "data": <EDT info in JSON schema>,
  "remark": {
    "ja": <remark in Japanese>,
    "en": <remark in English>
  },
  "note": {
    "ja": <note for DD in Japanese>,
    "en": <note for DD in English>
  }
}
```

Keyword	Type	Required	Description	Example
epc	string	yes	EPCコードを16進数表記したもの	0x80
validRelease	object	yes	property の記述が有効なAppendixのリリースバージョンの範囲	
validRelease.from	string	yes	有効範囲の始まりのAppendix	A, D
validRelease.to	string	yes	有効範囲の終わりのAppendix	L, latest (*1)
propertyName	object	yes	Appendix で定義された プロパティ名と property name	
propertyName.ja	string	yes	プロパティ名：日本語	動作状態

Keyword	Type	Required	Description	Example
propertyName.en	string	yes	プロパティ名：英語	Operation status
shortName (*2)	string	yes	property name の short name	
accessRule (*3)	object	yes	access rule	
accessRule.get	string	yes	Get の実装	required
accessRule.set	string	yes	Set の実装	optional
accessRule.inf	string	yes	状変時アナウンスの実装	required
descriptions	object	no	Appendix に記述された プロパティ内容と contents of property	
descriptions.ja	string	no	プロパティの説明：日本語	
descriptions.en	string	no	プロパティの説明：英語	
atomic (*4)	string	no	atomic operation が必要な場合のプロパティ	
data	object	yes	プロパティの値のデータタイプ。詳細は第４章を参照	
remark	object	no	備考および参考情報	
remark.ja	string	no	備考および参考情報：日本語	
remark.en	string	no	備考および参考情報：英語	
note (*2)	object	no	DD用の参考情報	
note.ja	string	no	DD用の参考情報：日本語	
note.en	string	no	DD用の参考情報：英語	

*1 latest は metaData.release の値。
*2 "shortName" と "note" は ECHONET Lite Web API の Device Description のためのデータ
*3 Appendix の「アクセスルール」と「必須」に関する記述と MRA の accessRule の関係を以下の表に示す。

アクセスルール	必須	accessRule
Get	blank	"get":"optional", "set":"notApplicable"
Get	○	"get":"required", "set":"notApplicable"
Set	blank	"get":"notApplicable", "set":"optional"
Set	○	"get":"notApplicable", "set":"required"

アクセスルール	必須	accessRule
Get/Set	blank	"get":"optional", "set":"optional"
Get/Set	○	"get":"required", "set":"required"
Get/Set	blank/○	"get":"optional", "set":"required"
Get/Set	○/blank	"get":"required", "set":"optional"

Appendix の「状変時アナウンス」に関する記述と accessRule の関係を以下の表に示す。

状変時アナウンス	accessRule
○	"inf":"required"
blank	"inf":"optional"

「必須」に関して条件がある場合、値として "required" の代わりに "required_c" を使用する。 機器オブジェクト詳細規定の「第１章本書の概要」に記述された「オプション必須」に該当する場合は、"required" の代わりに "required_o" の値を使用する。

*4 例えば、スマート電力量メータ (EOJ=0x0288) において、Get EPC=0xE2 を実行する前に, Set EPC=0xE5 を実行する必要がある。このようにセットで実行が必要なものを atomic operationと呼ぶ。この場合、プロパティ (EPC=0xE2) の atomic に "0xE5" を記述する。

Example


```
{
  "epc": "0x80",
  "validRelease": {
    "from": "A",
    "to": "latest"
  },
  "propertyName": {
    "ja": "動作状態",
    "en": "Operation status"
  },
  "shortName": "operationStatus",
  "accessRule": {
    "get": "required",
    "set": "required",
    "inf": "required"
  },
  "data": {
    "type": "state",
    "size": 1,
    "enum": [
      {
        "edt": "0x30",
```

```
        "name": "true",
        "descriptions": {
          "ja": "ON",
          "en": "ON"
        }
      },
      {
        "edt": "0x31",
        "name": "false",
        "descriptions": {
          "ja": "OFF",
          "en": "OFF"
        }
      }
    ]
  }
}
```

同一 EPC に対する複数の定義に関して

同一 EPC に対して、APPENDIXのリリースによってプロパティの定義が異なる場合、それぞれの property description object を配列の中に列挙する。以下の例では、EPC:0xE1 において、Release A-C と Release D 以降では、data schema の内容が異なる。

Example

```
"elProperties": [
  {
    "epc":"0xE1",
    "validRelease": {
      "from": "A",
      "to": "C"
    },
    "propertyName": {
      "ja": "開度レベル設定", ...
    },
    "shortName": "...",
    "accessRule": {...
    },
    "data": {
      "type": "level",
      "base": "0x31",
      "maximum": 8
    }
  },
  {
    "epc":"0xE1",
    "validRelease": {
```

```
        "from": "D",
        "to": "latest"
      },
      "propertyName": {
        "ja": "開度レベル設定", ...
      },
      "shortName": "...",
      "accessRule": {...
      },
      "data": {
        "type": "number",
        "format": "uint8",
        "unit": "%",
        "minimum": 0,
        "maximum": 100
      }
    }
  ]
```

4. Data type of property value

ECHONET Data (EDT) は、プロパティの値であり、バイナリーデータである。値の意味はプロパティごとに定義されている。
例：EDT=0x30 の場合

- Number: 48
- State: ON
- Level: 1

例：EDT=0xFFFF の場合

- Number: 65535 (2byte unsigned)
- Number: -1 (2byte signed)
- Number: 255 and 255 (two 1byte data)
- Number: -1 and -1 (two 1byte data)
- State: Over Flow

プロパティ値の意味を明確に記述するために、MRA では JSON schema の記述方法を独自に拡張して利用している。以下に MRA で使用しているデータタイプを示す。

Data Type	Description
number	数値
state	状態
numericValue	コード化された数値
level	レベル値

Data Type	Description
date	年月日
date-time	日時
time	時刻／時間
raw	raw データ
bitmap	ビットマップ
array	配列
object	（JOSN定義での）オブジェクト
oneOf	列挙された記述のうちのどれか一つ

- Data type "number", "state", "numericValue", "level", "date-time", "date", "time" を primitive type と呼ぶ。
- Data type "bitmap", "array", "object", "oneOf" を structured type と呼ぶ。
- Data type "raw" は、そのほかのデータタイプで表現できないデータである
- Data type "bitmap" は 数値や状態を bitmap で記述する
- Data type "array", "object", "oneOf" はデータ構造を記述する

4.1 number

データタイプ number は、数値を表す。ECHONET Lite は整数値または固定小数値を扱う。number のデータフォーマットは以下の通り。

int8, int16, int32, uint8, uint16, uint32

ECHONET Lite の仕様（第２部6.2.2）では、数値データの特定の値を Underflow や Overflow を表す code として定義している。

Data format

data format	range	Underflow code	Overflow code
int8	-127 to 126	0x80	0x7F
int16	-32767 to 32766	0x8000	0x7FFF
int32	-2147483647 to 2147483646	0x80000000	0x7FFFFFFF
uint8	0 to 253	0xFE	0xFF
uint16	0 to 65533	0xFFFE	0xFFFF
uint32	0 to 4294967293	0xFFFFFFFE	0xFFFFFFFF

Format

```
{
  "type": "number",
  "format": <format of number>,
  "unit": <unit>,
  "minimum": <minimum number>,
  "maximum": <maximum number>,
  "enum": <array of possible values>,
  "multiple": <multiple value>,
  "multipleOf": <minimum digit>,
  "coefficient": [<EPC of coefficient>
  ],
  "overflowCode": <true or false>,
  "underflowCode": <true or false>,
  "descriptions": {
    "ja": <description in Japanese>,
    "en": <description in English>
  }
}
```

Schema definitions

Keyword	Type	Required	Description
type	string	yes	"number"
format	string	yes	"int8", "int16", "int32", "uint8", "uint16", "uint32"
unit	string	no	単位
minimum	number	no	minimum number (*1)
maximum	number	no	maximum number (*1)
enum	array	no	特定の値のみ利用する場合は、値を列挙する
multiple	number	no	係数 (*2)
multipleOf	number	no	数値データのステップ (*3)
coefficient	array	no	係数として使用する EPCs (16進数表記 string) (*4)
overflowCode	boolean	no	overflow code 利用のflag. default は true. (*5)
underflowCode	boolean	no	underflow code 利用のflag. default は true. (*5)
descriptions	object	no	descriptions
descriptions.ja	string	no	description of the number in Japanese
descriptions.en	string	no	description of the number in English

*1 値は EDT の値を記述する
*2 multiple は、小数値を表現する場合にも使われる

> 12.5 = 125(EDT value) X 0.1(multiple)

*3 例："multipleOf":2 の場合は 0,2,4...。multiple が指定されている場合は乗算をする前の値のステップ。

*4 例：低圧スマート電力量メータ(EOJ:0x0288)の積算電力量計測値（EPC:0xE0）の場合、EPC 0xD3 と 0xE1 の値を係数として使用する。

> "coefficient":["0xD3", "0xE1"]

*5 Overflow や Underflow として定義されている値を数値として使う場合は、以下のように明示する

> "overflowCode":false
> "underflowCode":false

Example

```
{
  "type": "number",
  "format": "int16",
  "unit": "A",
  "minimum": -9999,
  "maximum": 9999,
  "multiple": 0.1,
  "coefficient": [
    "0xA0",
    "0xA1"
  ]
}
```

4.2 state

データタイプ "state" は状態を表す。例 0x30:ON, 0x31:OFF, 0x41:暖房.

Format

```
{
  "type": "state",
  "size": <data size>,
  "enum": [
    {
      "edt": <EDT value>,
      "name": <name of the state>,
      "descriptions": {
```

```
          "ja": <description in Japanese>,
          "en": <description in English>
        },
        "readOnly":true
      },
      ...
    ]
  }
```

Schema definitions

Keyword	Type	Required	Description
type	string	yes	"state"
size	number	yes	データサイズ（バイト数）を示す。bitmap の場合は 0 とする。
enum	array	yes	state object を列挙する

Schema definitions (state object)

Keyword	Type	Required	Description
edt	string	yes	16進数表記 (string) の EDT 値
name (*1)	string	yes	状態の名前
descriptions	object	yes	状態の説明
descriptions.ja	string	yes	状態の説明（日本語）
descriptions.en	string	yes	状態の説明（英語）
readOnly (*2)	boolean	no	read only flag. default は false.

*1 "name" は ECHONET Lite Web API の Device Description のために追加した。
*2 Get のレスポンスとしては利用されるが、Set の値としては利用できない場合に readOnly を true にする。例：オーブンレンジ (0x03B8) の庫内温度設定値 (0xE3) におけるプロパティ値, 0x8002: 未設定"。なお、アクセスルールが Get のみのプロパティでは、readOnly を利用しない。

Example

```
"data": {
  "type": "state",
  "size": 1,
  "enum": [
    {
      "edt": "0x41",
      "name": "true",
      "descriptions": {
```

```
          "ja": "沸き上げ中",
          "en": "Heating"
        }
      },
      {
        "edt": "0x42",
        "name": "false",
        "descriptions": {
          "ja": "非沸き上げ中",
          "en": "Not heating"
        }
      }
    ]
  }
```

4.3 numericValue

データタイプ "numericValue" は、ある値を別の値にマッピングする。例 0x00 は 1、0x01 は 10、0x02 は 100。

Format

```
  {
    "type": "numericValue",
    "size": <data size>,
    "enum": [
      {
        "edt": <EDT value>,
        "numericValue": <numeric value>
      },
      ...
    ]
  }
```

Schema definitions

Keyword	Type	Required	Description
type	string	yes	"numericValue"
size	number	yes	データサイズ（バイト数）を示す。bitmap の場合は 0 とする。
enum	array	yes	numericValue object を列挙する

Schema definitions (numericValue object)

Keyword	Type	Required	Description

Keyword	Type	Required	Description
edt	string	yes	16進数表記 (string) の EDT 値
numericValue	number	yes	数値

Example

```
{
  "type": "numericValue",
  "size": 1,
  "enum": [
    {
      "edt": "0x00",
      "numericValue": 1
    },
    {
      "edt": "0x01",
      "numericValue": 0.1
    },
    {
      "edt": "0x02",
      "numericValue": 0.01
    },
    {
      "edt": "0x03",
      "numericValue": 0.001
    },
    {
      "edt": "0x04",
      "numericValue": 0.0001
    }
  ]
}
```

4.4 level

データタイプ "level" はレベルを示す。例 家庭用エアコン (EOJ:0x0130) の風量設定 (EPC:0xA0) は８段階のレベル (1:弱 ... 8:強) をそれぞれ 0x31 から 0x38 の値で表している.

Format

```
{
  "type": "level",
  "base": <base value in Hex string>,
  "maximum": <maximum number of level>,
  "descriptions":{
```

```
      "ja": <description in Japanese>,
      "en": <description in English>
    }
  }
```

Schema definitions

Keyword	Type	Required	Description
type	string	yes	"level"
base	string	yes	レベル 1 に対応する EDT 値の16進数表記（string）の EDT 値
maximum	number	yes	レベルの最大値
descriptions	object	no	レベルの説明（*1）
descriptions.ja	string	no	レベルの説明（日本語）
descriptions.en	string	no	レベルの説明（英語）

*1 一つのプロパティで oneOf を使って複数のレベルが定義された場合などに、レベルの説明は有用である。

Example : 8 levels by 0x31 to 0x38

```
{
  "type":"level",
  "base":"0x31",
  "maximum":8
}
```

4.5 date

データタイプ "date" は 年月日（YYYYMMDD）を示す。年は２バイト、月と日は１バイトの数値である。年月（YYYYMM）のデータなど上位から一部分の要素を利用する場合は、"size" でデータサイズを指定する。データサイズを指定しない場合は 4バイトの年月日データである。

Data Format

item	data size	minimum	maximum
Year	2	1	9999
Month	1	1	12
Day	1	1	31

Format

```
{
  "type":"date",
  "size": <data size>
}
```

Schema definitions

Keyword	Type	Required	Description
type	string	yes	"date"
size	number	no	データサイズ（バイト数）を示す。

Example:

```
{
  "type":"date"
}
```

4.6 date-time

データタイプ "date-time" は 年月日時分秒（YYYYMMDDHHMMSS）を示す。年は２バイト、それ以外の要素は１バイトの数値である。上位から一部分の要素を利用する場合は、"size" でデータサイズを指定する。データサイズを指定しない場合は 7バイトの年月日時分秒データである。データサイズが４バイト以下の場合は、データタイプ "date" を使用する。

Data Format

item	data size	minimum	maximum
Year	2	1	9999
Month	1	1	12
Day	1	1	31
Hour	1	0	23
Minute	1	0	59
Second	1	0	59

データサイズは上位データからのサイズを示す。

Format

```
{
  "type":"date-time",
  "size": <data size>
}
```

Schema definitions

Keyword	Type	Required	Description
type	string	yes	"date-time"
size	number	no	データサイズ（バイト数）を示す。

Example: YYYY:MM:DD:hh:mm:ss

```
{
  "type":"date-time"
}
```

Example: YYYY:MM:DD:hh:mm (EOJ:0x0288, EPC:0xEC)

```
{
  "type":"date-time",
  "size":6
}
```

4.7 time

データタイプ "time" は 時刻または時間を示す時分秒（HHMMSS）を示す。それぞれの要素は１バイトの数値である。時分（HHMM）のデータなど上位から一部分の要素を利用する場合は、"size" でデータサイズを指定する。データサイズを指定しない場合は3バイトの時分秒データである。Hourの最大値は 23（default）であるが、maximumOfHour を利用することで、duration などの場合に最大値を 24 以上に設定することも可能である。

Data Format

item	data size	minimum	maximum
Hour	1	0	23 (or 255)
Minute	1	0	59
Second	1	0	59

Format

```
{
  "type":"time",
  "size": <data size>,
  "maximumOfHour": < maximum value of Hour >
}
```

Schema definitions

Keyword	Type	Required	Description
type	string	yes	"time"
size	number	no	データサイズ（バイト数）を示す。
maximumOfHour	number	no	Hour の最大値を設定する。default は 23 である。

Example: HH:MM:SS

```
{
  "type":"time"
}
```

Example: HH(0 to 255):MM

```
{
  "type":"time",
  "size":2,
  "maximumOfHour":255
}
```

4.8 bitmap

データタイプ "bitmap" は bitmap を利用して値や状態を表す。

例： 全体で４バイト。１バイト目のデータのMSBと２バイト目のデータの下位4bitsを利用する。利用するbitmapの位置は index と bitmask で指定する。index は何番目のバイトであるかを指定し、bitmask はそのバイトの中の bit の位置を指定する。

Example:

```
    |<----------- 4 bytes ----------->|
    index:0  index:1  index:2  index:3
    XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
    MSB  LSB MSB  LSB MSB  LSB MSB  LSB

    X******* ****XXXX ******** ********
    ^            ^^^^
    |            ||||
    |            ++++--- index = 1, bitmask = 0b00001111, data = 0...15
    +--- index = 0, bitmask = 0b10000000, data = 0 or 1
```

Format

```
{
  "type": "bitmap",
  "size": <size of total bitmaps data in bytes>,
  "bitmaps": [
    {
      "name": <bitmap name>,
      "descriptions": {
        "ja": <description of bitmap in Japanese>,
        "en": <description of bitmap in English>
      },
      "position": {
        "index": <index of the byte of a bitmap>,
        "bitMask": <bitmask to specify effective bits>
      },
      "value": <data type object>
    },
    ...
  ]
}
```

Schema definitions

Keyword	Type	Required	Description
type	string	yes	"bitmap"
size	number	yes	全体のデータサイズ（バイト数）を示す。
bitmaps	array	yes	bitmap object の列挙

Schema definitions (bitmap object)

Keyword	Type	Required	Description

Keyword	Type	Required	Description
name	string	yes	bitmap 名 (*1)
descriptions	object	yes	bitmap の説明
descriptions.ja	string	yes	bitmap の説明（日本語）
descriptions.en	string	yes	bitmap の説明（英語）
position	object	yes	bitmap の位置
position.index	number	yes	何バイト目かを示す
position.bitMask	string	yes	該当するバイトの中のbitmapの１を示す
value	object	yes	bitmapで表現する値のデータタイプオブジェクト

*1 "name" は ECHONET Lite Web API の Device Description のために追加した。
*2 データタイプが state の場合, state のサイズは 0 とする。

Example: 電気温水器 (EOJ:026B) 警報発生状態(EPC:0xC2)

> First byte, bit0: Out of hot water
> First byte, bit1: Water leaking
> First byte, bit2: Freezing


```
{
  "type": "bitmap",
  "size": 4,
  "bitmaps": [
    {
      "name": "noHotWater",
      "descriptions": {
        "ja": "湯切れ警報",
        "en": "Out of hot water"
      },
      "position": {
        "index": 0,
        "bitMask": "0b00000001"
      },
      "value": {
        "type": "state",
        "size": 0,
        "enum": [
          {
            "edt": "0x01",
            "name": "true",
            "descriptions": {
              "ja": "発生",
              "en": "Alarm"
```

```
            }
          },
          {
            "edt": "0x00",
            "name": "false",
            "descriptions": {
              "ja": "正常",
              "en": "No Alarm"
            }
          }
        ]
      }
    },
    {
      "name": "waterLeaking",
      "descriptions": {
        "ja": "漏水警報",
        "en": "Water leaking"
      },
      "position": {
        "index": 0,
        "bitMask": "0b00000010"
      },
      "value": {
        "type": "state",
        "size": 0,
        "enum": [
          {
            "edt": "0x01",
            "name": "true",
            "descriptions": {
              "ja": "発生",
              "en": "Alarm"
            }
          },
          {
            "edt": "0x00",
            "name": "false",
            "descriptions": {
              "ja": "正常",
              "en": "No Alarm"
            }
          }
        ]
      }
    },
    {
      "name": "waterFreezing",
      "descriptions": {
        "ja": "凍結警報",
        "en": "Water frozen"
```

```
        },
        "position": {
          "index": 0,
          "bitMask": "0b00000100"
        },
        "value": {
          "type": "state",
          "size": 0,
          "enum": [
            {
              "edt": "0x01",
              "name": "true",
              "descriptions": {
                "ja": "発生",
                "en": "Alarm"
              }
            },
            {
              "edt": "0x00",
              "name": "false",
              "descriptions": {
                "ja": "正常",
                "en": "No Alarm"
              }
            }
          ]
        }
      }
    ]
  }
```

4.9 raw

データタイプ "raw" は、他のデータタイプで表現できないデータを示す。このデータタイプではサイズのみ記述する。データ値の詳細はAPPENDIX (PDF) を参照。

Format

```
{
  "type":"raw",
  "minSize": < minimum data size >,
  "maxSize": < maximum data size >
}
```

Schema definitions

Keyword	Type	Required	Description

Keyword	Type	Required	Description
type	string	yes	"raw"
minSize	number	yes	データサイズの最小値
maxSize	number	yes	データサイズの最大値

Example: 8 byte data

```
{
  "type":"raw",
  "minSize":8,
  "maxSize":8
}
```

Example: variable size data from 1 to 8 byte

```
{
  "type":"raw",
  "minSize":1,
  "maxSize":8
}
```

4.10 array

データタイプ "array" は項目の列挙を示す。それぞれの項目は同一のデータタイプである。項目のデータタイプは primitive type (number, state, ...), または structured type (bitmap, array, object, oneOf) である。データサイズは、項目のデータサイズとデータ個数の乗算となる。

Format

```
{
  "type":"array",
  "itemSize": < data size of each item>,
  "minItems": < minimum number of items>,
  "maxItems": < maximum number of items>,
  "items": <data type object of each item>
}
```

Schema definitions

Keyword	Type	Required	Description

Keyword	Type	Required	Description
type	string	yes	"array"
itemSize	number	yes	項目のデータサイズ
minItems	number	no	項目の最小数
maxItems	number	yes	項目の最大数
items	object	yes	項目のデータタイプオブジェクト

Example

```
{
  "type":"array",
  "itemSize":2,
  "maxItems":24,
  "items":{
    "type":"number",
    "format":"int16",
    "unit":"A",
    "minimum":-9999,
    "maximum":9999,
  }
}
```

4.11 object

データタイプ "object" は、複数の要素からなる構造化されたデータを示す。各要素のデータタイプは primitive type (number, state, ...), または structured type (bitmap, array, object, oneOf) である。データサイズは、各要素のデータサイズの合計である。

Format

```
{
  "type": "object",
  "properties": [
    {
      "elementName": {
        "ja": <element name in Japanese>,
        "en": <element name in English>
      },
      "shortName": <element name>,
      "element": <data type object>
    },
    ...
```

```
    ]
  }
```

Schema definitions

Keyword	Type	Required	Description
type	string	yes	"object"
properties	array	yes	要素の列挙
elementName	object	yes	要素名
elementName.jp	string	yes	要素名（日本語）
elementName.en	string	yes	要素名（英語）
shortName (*1)	string	yes	要素の short name
element	object	yes	要素のデータタイプオブジェクト

(*1) "shortName" は ECHONET Lite Web API の Device Description のために追加した。

Example: 一般照明, EPC=0xC0

```
{
  "type": "object",
  "properties": [
    {
      "elementName": {
        "ja": "R",
        "en": "R"
      },
      "shortName": "red",
      "element": {
        "type": "number",
        "format": "uint8",
        "minimum": 0,
        "maximum": 255
      }
    },
    {
      "elementName": {
        "ja": "G",
        "en": "G"
      },
      "shortName": "green",
      "element": {
        "type": "number",
        "format": "uint8",
```

```
          "minimum": 0,
          "maximum": 255
        }
      },
      {
        "elementName": {
          "ja": "B",
          "en": "B"
        },
        "shortName": "blue",
        "element": {
          "type": "number",
          "format": "uint8",
          "minimum": 0,
          "maximum": 255
        }
      }
    ]
  }
```

Example: 低圧スマート電力量メータ, EPC=0xE2

```
  {
    "type": "object",
    "properties": [
      {
        "elementName": {
          "ja": "積算履歴収集日",
          "en": "day for which the historical data of measured cumulative amounts of
  electric energy is to be retrieved"
        },
        "shortName": "day",
        "element": {
          "type": "number",
          "format": "uint8",
          "unit": "day",
          "minimum": 0,
          "maximum": 99
        }
      },
      {
        "elementName": {
          "ja": "積算電力量計測値",
          "en": "measured cumulative amounts of electric energy"
        },
        "shortName": "energy",
        "element": {
          "type": "array",
          "itemSize": 4,
```

```
          "minItems": 48,
          "maxItems": 48,
          "items": {
            "type": "number",
            "format": "uint32",
            "unit": "kWh"
          }
        }
      }
    ]
  }
```

4.12 Mixed type: oneOf

データタイプの定義がプロパティ値の範囲によって異なる場合、oneOf という keyword でデータタイプオブジェクトを列挙する。

Format

```
{
  "oneOf":[
    <data type object>,
    <data type object>,
    ...
  ]
}
```

Example: 0x00 to 0x28:number, 0x31 to 0x38:level, 0x41:state

```
{
  "oneOf": [
    {
      "type": "number",
      "format": "uint8",
      "unit": "Celsius",
      "minimum": 0,
      "maximum": 40
    },
    {
      "type": "level",
      "base": "0x31",
      "maximum": 8
    },
    {
      "type": "state",
      "size": 1,
```

```
        "enum": [
          {
            "edt": "0x41",
            "name": "auto",
            "descriptions": {
              "ja": "自動",
              "en": "Automatic"
            }
          }
        ]
      }
    ]
  }
```

5. definitions

例えば、家庭用エアコンの湿度計測値（EPC:0xBA）や電動ブラインドの開度レベル設定（0xE1）では、１バイトのデータで 0 - 100% の値を表現する。MRA ではどちらの場合も以下の schema を記述する。

```
"data":{
  "type":"number",
  "format":"uint8",
  "unit":"%",
  "minimum":0,
  "maximum":100
}
```

家庭用エアコンと電動ブラインドのそれぞれの MRA で上記の schema を記述するよりも、１バイトのデータで 0 - 100% の値を表現する schema をテンプレートとして別途定義し、MRA ではその参照を記述することで、間違いを防ぎデータ量を削減することができる。definitions.json にはこのようなテンプレートを記述し、MRA では keyword "$ref" を使って参照する。

format of definition.json

```
{
  "definitions":{
    <template name>:{ <schema> },
    ...
    }
}
```

Example of definitions.json

```
{
  "definitions": {
    "number_0": {
      "type": "number",
      "format": "uint8",
      "minimum": 0,
      "maximum": 0
    },
    "number_0-100percent": {
      "type": "number",
      "format": "uint8",
      "unit": "%",
      "minimum": 0,
      "maximum": 100
    },
    ...
  }
}
```

Example of 0x0130.json

```
{
  "eoj": "0x0130",
  "validRelease": {
    "from": "A",
    "to": "latest"
  },
  "className": {
    "ja": "家庭用エアコン",
    "en": "Home air conditioner"
  },
  "shortName": "homeAirConditioner",
  "elProperties": [
    {
      "epc": "0xBA",
      "validRelease": { ... },
      "propertyName": {
        "ja": "室内相対湿度計測値",
        "en": "Measured value of room relative humidity"
      },
      "accessRule": { ... },
      "descriptions": { ... },
      "data": {
        "$ref": "#/definitions/number_0-100percent"
      }
    },
```

```
    ]
  }
```

なお、上記例の "$ref": "#/definitions/xxxx" の PATHは、MRA として提供するファイルのデータを、以下の構成の JSON データとして利用する場合を想定したものである。

```
{
  "metaData": { < metaData.json の内容 > },
  "definitions" : { < definitions.json の内容 > },
  "nodeProfile" : { < nodeProfile.json の内容 > },
  "superClass" : { < superClass の内容 > },
  "devices" : [
    { < 0x0130.json の内容 > },
    ...
    { < 0x0290.json の内容 > }
    ...
  ]
}
```

6. Node profile: 0x0EF0

Node profile の JSON data は、node profile の super class の情報と node profile の情報の両方を記述する。

7. Super class

Super class の JSON data は、機器オブジェクトの super class の情報を記述する。