# Espressif IoT SDK:
# Json Naming Guidelines

| | |
|---|---|
| **Status** | Released |
| **Current version** | V0.1 |
| **Author** | Han Liu |
| **Completion Date** | 2014.6.19 |
| **Reviewer** | Jiangang Wu |
| **Completion Date** | 2014.6.19 |

**[    ] CONFIDENTIAL**
**[    ] INTERNAL**
**[ √ ] PUBLIC**

# Version Info

| Date | Version | Author | Comments/Changes |
|---|---|---|---|
| 2014.3.7 | 0.1 | Han Liu | Draft |
| | | | |
| | | | |
| | | | |

**Disclaimer and Copyright Notice**

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member Logo is a trademark of the Wi-Fi Alliance.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

# Table of Contents

Version Info ........................................................................................................2
Table of Contents ..................................................................................................3
1. Foreword .........................................................................................................4
2. Guidelines.........................................................................................................5
2.1. General Guidelines............................................................................................5
2.1.1. Comments.....................................................................................................5
2.1.2. Use of Double Quotation Marks ......................................................................5
2.1.3. Flat data VS Structured Data ..........................................................................5
2.2. Property Guidelines ..........................................................................................6
2.2.1. Naming Properties ........................................................................................6
2.2.2. Naming Conflict.............................................................................................7
2.2.3. Property values .............................................................................................8

# 1. Foreword

This introduces to you some instructional guidelines and suggestions on the establishment of JSON APIs based on ESP_IOT lib. In general, JSON APIs shall comply with standards on JSON.org. The principles mentioned in this document specifies and standardizes some specific situations, therefore, they are applicable to REST styled JSON request and response of API.

# 2. Guidelines

The following items should be declared:

Property – name/value pair (JSON)

Property name – name of the property

Property value – value that is designated to property

Example:

{

// a pair of name/value can be regarded as an “property”.

" property Name": " property Value"

}

## 2.1. General Guidelines

### 2.1.1. Comments

Comments should not be included in JSON properties.

### 2.1.2. Use of Double Quotation Marks

If a certain property value needs to be quoted, double quotation marks must be used. All string values must be within a pair of double quotation marks. All other types such as Boolean values or numbers must not muse double quotation marks.

### 2.1.3. Flat data VS Structured Data

The flat data structure is used in JSON here, and data should not be randomly grouped.

However, in some occasions, structured data is more meaningful to developers. For example, a set of properties used to describe the whole structure is useful for it is used to maintain the layers of the data. Attention shall be paid to such occasions.

Flat Example:

```
{
"ssid": "tenda_837R",
"password": ""
}
```

Structured Example:

```
{
"Request": {
"Station": {
“Connect":{
"ssid": "tenda_837R",
"password": ""
}
}
}
}
```

## 2.2. Property Guidelines

### 2.2.1. Naming Properties

Property names with certain meanings should be chosen, and the following guidelines abided:

- Property name must be a name that can be used to define semantic meanings;
- Property name must be CamelCase ASCII strings;
- The first character must not include numbers;
- If an object property has a child, but no value, then the first character must be capitalized;

- If an object property has no child, but has value, then the first character must not be capitalized;
- The remaining characters can include numbers.

Example:

```
{
"Response":{
"status":0
}
}
```

## 2.2.2. Naming Conflict

New properties can be added to the existing list. If the naming of the properties conflicts with others, you can choose a new property name or version to solve this problem.

Example:

```
{
"version": "1.0",
"message": {
"sucessful": true,
"information": "flash.bin",
"data": ["sensor", "device"]
}
}
```

To make “information” a reserved word, do either one of the following two:

- Choose a different name

```
{
"version": "1.0",
"Message": {
"successful": true,
"information": "flash.bin",
"information0": "irom.bin",
"Data": ["sensor", "device"]
}
}
```

- Rename the property in a newer version.

```
{
"version": "2.0",
"Message": {
"successful": true,
"information": "irom.bin",
"Data": ["sensor", "device"]
}
}
```

### 2.2.3. Property values

JSON.org clearly specify what kinds of data can be used as property value. The property value must be Unicode Boolean, number, string, object, array, or null, etc.

JavaScript expressions are unacceptable. APIs should follow the JSON guidelines, and can choose the most appropriate type of data for each object property.

Example:

```
{
"message": null,   // null
"sucessful": false,      // boolean
"status": 42,            // number
"name": "Bart",    // string
"response": {},    // object
"data": []         // array
}
```