Message Queuing Telemetry Transport（MQTT）

欢迎访问设备云门户网站<http://open.iot.10086.cn/>注册用户，获取最新文档。

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订日期** | **修订内容** | **说明** |
| V1.1 | 2015/9/8 | 草稿 |  |
|  |  |  |  |
|  |  |  |  |

目录

[Message Queuing Telemetry Transport（MQTT） 1](#_Toc430682802)

[1 说明 1](#_Toc430682803)

[2 设备与业务接入模式 1](#_Toc430682804)

[3 接入流程（待更新） 2](#_Toc430682805)

[4 消息格式 2](#_Toc430682806)

[5 消息差异及说明 2](#_Toc430682807)

[5.1 CONNECT 2](#_Toc430682808)

[5.1.1 差异 2](#_Toc430682809)

[5.1.2 说明 3](#_Toc430682810)

[5.2 PUBLISH 3](#_Toc430682811)

[5.2.1 差异 3](#_Toc430682812)

[5.2.2 说明 4](#_Toc430682813)

[5.3 SUBSCRIBE 8](#_Toc430682814)

[5.3.1 差异 8](#_Toc430682815)

[5.3.2 说明 9](#_Toc430682816)

[5.4 UNSUBSCRIBE 9](#_Toc430682817)

[5.5 加密 9](#_Toc430682818)

[6 命令请求 9](#_Toc430682819)

# 说明

MQTT协议详细内容请参见MQTT version 3.1.1官方文档，本文档对此不做详细说明，仅指明OneNet的要求、默认参数、以及当前实现与MQTT官方文档的差异。

# 设备与业务接入模式

**设备层：**利用平台提供的MQTT SDK或者第三方SDK实现MQTT客户端，用于上报业务数据点到平台。

**业务应用层：**若要自定义实现业务平台，可通过HTTP 协议的RESTful API操作OnetNet提供的资源（设备、数据点、命令控制等资源的增删查改）。



适用场景：在充分分析业务数据模型的基础上，认为OneNet提供的设备-数据流-数据点模型适合业务数据存储。优先推荐新业务使用该模式。

# 接入流程

* 1. 访问平台<http://open.iot.10086.cn/>注册用户；
  2. 用户根据业务情况，在“连接请求”章节中选择EDP登录方式（目前公测阶段，页面还未提供MQTT登录选项，登录方式与EDP兼容）；
  3. 根据登录方式，填写设备相关属性，在项目下新增设备，获取项目ID、设备ID，以及api-key等信息；
  4. 设备发送TCP连接请求到以下地址，发送封装的报文与平台交互。

**平台服务器地址183.230.40.39，TCP 端口6002**

# 消息格式

详见MQTT version 3.1.1 官方文档。

# 消息差异及说明

本节主要描述与标准的MQTT version 3.1.1相比，当前有哪些功能不支持或者有差异，以及为了与整个平台整合，对某些字段所做的要求，比如鉴权信息应该如何填写等。

## CONNECT

### 5.1.1 差异

1. **user flag与password flag**

平台不允许匿名登陆，因此这两个标志位在连接时必须设置为1，否则认为协议错误，平台将会断开连接。

1. **will flag**

如果设置了will flag，在设备异常断开后，平台会分发will message给所有订阅了此消息的设备。Will message会被当作二进制字符串发布，即第[5.2.2节](#_5.3.2_说明)中普通topic下的二进制字符串类型。

平台将以用户订阅时指定的qos来发送will message，而不是will qos中指定的qos，因为平台认为用户更了解这条消息对自己是否重要。

如果设置了will flag，且设备连接时将will qos设置为0，则其与平台断连后，平台会发送此will message，但是不会保存此will message，并且如果有已经保存的will message，也会将其丢弃。

### 5.1.2 说明

1. **版本**

平台只支持版本v 3.1.1，不支持更老的版本。使用第三方客户端时需要注意选择正确的版本。

1. **session保存时间**

若客户端将clean session标志位设置为0，当其断开后，平台将会保存session，这在弱网环境非常有用，因为网络不稳定可能导致设备频繁与服务器断开连接。平台保存session的时间为一小时，也就是说如果一小时内设备未能再次成功连接，则平台将会丢弃与此session相关的所有信息。

另外，平台拥有多台接入机，session的保存与恢复暂时不会跨越接入机。如果设备下次登陆的接入机与上次登陆的接入机不一致，则登陆后，平台不会恢复session，连接响应的sp标志位会被按要求置为0。设备应该在断连后重新连接时正确处理此标志，执行重新订阅所需要的topic等操作。

1. **保活时间**

平台要求最大保活时间为240秒，0将会被自动转换为最大保活时间。设置大于240秒的保活时间会被认为协议错误，平台将会断开连接。

按照MQTT协议的要求，实际的保活时间将会是此值的1.5倍，也即最大保活时间为6分钟。

1. **鉴权**

与鉴权相关的字段包含client id，username和password，支持两种鉴权方式。

|  |  |  |
| --- | --- | --- |
| **登陆认证方式** | **字段设置** | **消息示例** |
| 1 | client\_id设置为“设备ID”  username设置为“项目ID”  password设置为“鉴权信息（api-key）” | client\_id=”12345”  username=”433223”  password=” Bs04OCJioNgpmvjRphRak15j7Z8=” |
| 2 | client\_id设置为 “auto”  username设置为“项目ID”  password设置为“鉴权信息（auth\_info）” | client\_id=”auto”  username=”433223”  password=”{"MAC":"MAC-100", "SN" : "SN-100" }” |

各字段说明如下：

设备ID：申请设备时平台返回的ID；

项目ID：在平台添加项目时平台生成的ID；

鉴权信息（api-key）：在平台申请的可以操作该设备的api-key字符串；

鉴权信息（auth\_info）：在平台申请设备时填写设备的auth\_info属性（json对象字符串），该属性需要项目内具备唯一性；

## PUBLISH

### 5.2.1 差异

1. **topic**

当发布数据点时，以$开始的为系统保留的topic，其它的为普通topic。系统保留的topic目前主要有三个，一个是“$SYS/cmdreq/cmd\_uuid”，用于命令执行，其中的cmd\_uuid部分对于每条命令都不一样，订阅时应该订阅“$SYS/cmdreq/#”，详见[第6节命令请求](#_命令请求)。另外两个用于一次发布多数据点，分别是“$SYS/savedata”和“$SYS/passdata”。区别是平台会存储前者携带的数据点，而不会存储后者携带的数据点。其它的系统topic将会被认为协议错误，平台将会断开连接。

普通topic一次只携带一个数据点信息。普通topic数据格式及系统topic的数据格式见第[5.2.2](#_5.2.2_说明)节。

1. **retain msg保存时间**

用户以qos不为0且设置了retain标记的方式publish的消息，平台会保存，但是不会永久保存，以防止恶意行为。当一个设备断开连接后，它所发布的retain消息还将继续保存一个小时。如果在一小时内设备重新连接上，则其retain消息将继续保存下去。

### 5.2.2 说明

本节主要讲解发布数据点时，不同类型的topic的payload的封装格式。

1. **普通topic**

不以‘$’开始的topic为普通topic，其topic名字为上报的数据流名称，payload要求按以下格式封装。

第一字节为payload类型头，第0-2位表示支持的数据类型，共可支持8种，目前支持6种；第3-4位系统保留。第5-6位表示每种类型的可能选项，第7位表示是否需要平台存储； 第二字节及其以后的字节为真正的数据，基本格式如下。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 平台是否存储 | 时间选项 | 其它选项 | 系统保留 | | 数据类型 | | |
| Byte 2 | 真正的payload | | | | | | | |

数据类型目前支持null，int32，int64，double，string， 二进制字符串和bool型等七种类型。

选项的第6位固定为时间选项，若不带时间选项，则平台存储时会利用系统当前时间作为数据点的发布时间存储。时间精确到毫秒，长度为8字节，按大端字节序存放。

选项的存储顺序按高位到低位的顺序。比如第6和第5位均被置上，则从第二字节开始，先存放的是8字节的时间信息，再存第5位选项的内容，最后存放实际的数据。

1. **NULL**

类型定义为0x0，没有数据。只支持时间选项，若第6位被置上1，表示payload中含有时间信息。这种类型的数据按普通数据处理，如果设置了retain标记，则会清除此topic对应的retain消息。

示例1：不带时间，也不需要平台存储，格式如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

示例2 ：带时间，不需要平台存储，格式如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte 2 | 8字节时间的最高位 | | | | | | | |
| Byte 3 | 8字节时间的其它位，按大端字节序存放 | | | | | | | |
| Byte 4 |
| Byte 5 |
| Byte 6 |
| Byte 7 |
| Byte 8 |
| Byte 9 | 8字节时间的最低位 | | | | | | | |

1. **Int32**

类型定义为0x2，占4个字节，按大端字节序存放。只支持时间选项，若第6位被置上1，表示payload中含有时间信息。

示例：需要平台存储，不带时间。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Byte 2 | 四字节int32类型的最高位 | | | | | | | |
| Byte 3 | 四字节int32类型的次高位 | | | | | | | |
| Byte 4 | 四字节int32类型的次低位 | | | | | | | |
| Byte 5 | 四字节int32类型的最低位 | | | | | | | |

1. **Int64**

类型定义为0x2，占8个字节，按大端字节序存放。只支持时间选项，若第6位被置上1，表示payload中含有时间信息。

示例：不需要平台存储，不带时间。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Byte 2 | 8字节int64类型的最高位 | | | | | | | |
| Byte 3 | 8字节int64类型的其它位，按大端字节序存放 | | | | | | | |
| Byte 4 |
| Byte 5 |
| Byte 6 |
| Byte 7 |
| Byte 8 |
| Byte 9 | 8字节int64类型的最低位 | | | | | | | |

1. **double**

类型定义为0x3，占8个字节，按大端字节序存放。只支持时间选项，若第6位被置上1，表示payload中含有时间信息。

示例：需要平台存储，带时间。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| Byte 2 | 8字节时间信息的最高位 | | | | | | | |
| Byte 3 | 8字节时间信息的其它位，按大端字节序存放 | | | | | | | |
| Byte 4 |
| Byte 5 |
| Byte 6 |
| Byte 7 |
| Byte 8 |
| Byte 9 | 8字节时间信息的最低位 | | | | | | | |
| Byte 10 | 8字节double类型的最高位 | | | | | | | |
| Byte 11 | 8字节double类型的其它位，按大端字节序存放 | | | | | | | |
| Byte 12 |
| Byte 13 |
| Byte 14 |
| Byte 15 |
| Byte 16 |  | | | | | | | |
| Byte 17 | 8字节double类型的最低位 | | | | | | | |

1. **string**

类型定义为0x4。可能选项为第6位。只支持时间选项，表示payload中含有时间信息。除去选项信息以外的数据为真正的字符串信息。如果这种字符串中含义‘\0’，则‘\0’后的数据将会被丢弃。

示例：不需要平台存储，不带时间，存储值为“HELLO”。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Byte 2 | ‘H’ | | | | | | | |
| Byte 3 | ‘E’ | | | | | | | |
| Byte 4 | ‘L’ | | | | | | | |
| Byte 5 | ‘L’ | | | | | | | |
| Byte6 | ‘O’ | | | | | | | |

1. **二进制字符串**

类型定义为0x5。可能选项为第6和第5位。

若第6位被置上1，表示payload中含有时间信息。若第5位被置上表示含有描述信息，描述信息用于对二进制数据的说明，表示方式为二字节的长度信息后跟描述字符串。除去选项信息以外的数据为真正的字符串信息。

示例：二进制字符串“bin”，需要平台保存，含时间信息，带描述字段“desc”。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
|  | 平台保存 | 带时间信息 | 含有描述字段 | 系统保留 | | 数据类型5，二进制字符串 | | |
| Byte 2 | 8字节时间信息的最高位 | | | | | | | |
| Byte3 | 8字节时间信息的其它位，按大端字节序存放 | | | | | | | |
| Byte4 |
| Byte5 |
| Byte6 |
| Byte7 |
| Byte8 |
| Byte9 | 8字节时间信息的最低位 | | | | | | | |
| 两位描述字符串长度 | | | | | | | | |
| Byte10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte11 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Byte12 | ‘d’ | | | | | | | |
| Byte13 | ‘e’ | | | | | | | |
| Byte14 | ‘s’ | | | | | | | |
| Byte15 | ‘c’ | | | | | | | |
| 二进制字符串内容 | | | | | | | | |
| Byte16 | ‘b’ | | | | | | | |
| Byte17 | ‘I’ | | | | | | | |
| Byte18 | ‘n’ | | | | | | | |

1. **Bool**

类型定义为0x06，占一个字节。只支持时间选项，若第6位被置上1，表示payload中含有时间信息。

示例：不需要平台存储，不带时间。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **字节** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Byte 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Byte 2 | 1字节的bool值 | | | | | | | |

注：对于bool类型，当bool值被设置为非0和1的数据，如果订阅设备与发送设备在同一个接入机上，则此值会按原样直接分发。如果不在同一个接入机上，由于中间数据格式的转换，最终将会被转换为1，而不是原值。

1. **topic为“$SYS/savedata”或者“$SYS/passdata”**

Client用这两个topic可同时发布多个数据点。对于前者的数据，平台会存储；对于后者的数据，平台只会转发给订阅者，不会存储。

报文PAYLOAD格式如下：

|  |  |
| --- | --- |
| BYTE1 | 格式类型(datapoint\_type)  1：json格式1，2：json格式2，3: 标号分隔字符串，4：二进制 |
| BYTE2…n | 数据点内容(datapoint\_content) |

1. **datapoint\_type = 1**

datapoint\_content 格式json字符串：

**{ “datastream\_id1”:”value1”, “datastream\_id2”:”value2”,…}**

datastream\_id不允许为空，如果为空会被认为协议错误，平台将会断开连接。

这种类型的消息不带时间信息，平台会把当前系统时间作为此条消息的发布时间存放在数据库中。

1. **datapoint\_type = 2**

datapoint\_content 格式json字符串：

**{ “datastream\_id1”:{“datetime1”:”value1”},“datastream\_id2”: {“datetime2”:”value2”},…}**

datastream\_id不允许为空，如果为空会被认为协议错误，平台将会断开连接。

datetime的格式为2015-03-22 22:31:12,22.5，如果格式错误，平台将会采用系统当前时间。

1. **datapoint\_type = 3**

datapoint\_content是英文分号间隔域（每个数据流），英文逗号间隔域内信息，格式如下：

**feild0;feild1;…;feildn**

其中，每个field格式支持3种：

|  |  |
| --- | --- |
| Field格式 | 说明 |
| Datastream\_id,datetime,value | 3个子字段，分别是数据流ID,时间戳，数据值 |
| Datastream\_id,value  Datastream\_id,,value | 2个子字段，省略时间戳，分别是数据流ID和数据值  2个子字段，省略时间戳，分别是数据流ID和数据值 |
| Value | 1个子字段，省略了数据ID和时间戳，只传输数据值，平台将用该域(feild)所在的位置号（从0开始）作为数据流ID。 |

示例：temperature,2015-03-22 22:31:12,22.5;102;humidity,35%;pm2.5,89;1001

如果域内数据项超时3个，会被认为协议错误，平台将会断开连接。可通过逗号个数简单判断域内数据项，如果域内有超过2个逗号，则认为协议错误，比如：“a,b,c,d”。

此类型datastream\_id可以为空，如果为空，此域将会为忽略，以支持省略数据流ID的域。比如想设置一个数据流ID为2，值为test的数据点，可以利用以下格式：“;;test”。同样，字符串“,a;;test”，因为第一个域“,a;”缺少数据流ID，此项将被忽略，它与“;;test”等价。

此类型的value若为空，则也会被跳过。比如“ds,time,;”，由于这里有两个逗号，说明第一项为数据流ID，第二项为时间，第三项为value，但是由于第三项为空，所以它将被忽略。

这种格式的value将会被当作字符串存储。

1. **datapoint\_type = 4**

datapoint\_content内容是二级制数据点，分为两个部分：

|  |
| --- |
| 2字节描述字符串长度指示 |
| 描述字符串内容：datastream\_id,datetime,description  其中，datatime和description可以省略。 |
| 4字节二进制数据内容长度指示 |
| 二进制数据内容(不操过4M) |

## SUBSCRIBE

### 5.3.1 差异

订阅通配符目前只支持“#”，不支持“+”；

### 5.3.2 说明

订阅普通topic格式为：userid/apikey/devid/topic，表示需要订阅某个设备的某个topic。其中前三项会被用于鉴权，鉴权失败则会按要求返回0x80，表示订阅失败。订阅成功后，若有相应订阅消息到来后，此消息会被分发给订阅者。消息分发的topic为devid/topic。Payload按照[第5.2.2节普通topic](#_5.2.2_说明)的封装格式封装。

订阅系统topic是针对当前设备，不需要鉴权，因此没有前面三项。目前系统topic的订阅只支持命令请求，格式为：$SYS/cmdreq/#，详见：[第6节命令请求](#_命令请求)。

注意：平台支持以json格式同时上传多数据点，json中支持的数组和对象类型在[5.2.2节普通topic](#_5.2.2_说明)封装格式中没有对接的格式，如果用户利用json上传这两种类型的数据时，最终会被格式化为[5.2.2节普通topic](#_5.2.2_说明)封装格式中的string类型发送给订阅设备。

平台向设备分发消息，对于qos为1或者2的消息，需要设备响应。如果设备未及时响应，平台会重发消息，重发的频率按指数增长，第一次间隔1秒，第二次间隔2秒，最长间隔64秒，累积时间不大于设备的保活时间。当超过设备的保活时间后，平台将丢弃此消息，因为在保活时间内，设备必须向平台发送消息，否则平台将断开与此设备的连接。如果平台在这段时间内能收到其它信息，但是一直收不到消息的响应，则认为是设备恶意不回此消息，帮将其丢弃。

## UNSUBSCRIBE

**5.4.1 说明**

取消订阅的topic格式为devid/topic。

## 加密

暂不支持。

# 命令请求

命令请求用于API向设备发送命令请求，接口详见《业务侧接口-RESTfulAPI.docx》“命令执行”章节。

当API向设备发送命令后，无论设备是否订阅了命令请求的topic，平台都会将此命令请求publish给设备。如果没有订阅收到的消息对应的topic，有些SDK可能会丢弃收到的消息，对于这类SDK建议在设备启动时订阅命令请求topic。

命令请求topic采用系统保留的topic前缀“$SYS/cmdreq”，因此若需订阅，订阅的topic为“$SYS/cmdreq/#”。注意普通topic的形式为userid/apikey/devid/topic，前三项用于鉴权，由于系统topic是针对当前订阅设备的，其不需要鉴权，所以格式上有区别。

如果不订阅，平台将以qos为1的方式向设备publish命令请求，否则qos为订阅时指定的qos。

平台将按照[5.2.2节中普通topic二进制字符串](#_5.2.2_说明)的格式向设备发送命令。设备收到的命令执行请求的topic形如“$SYS/cmdreq /cmd\_uuid”，如果设备需要将此命令执行的结果上传平台，以供平台存储及后续查询，则需要向平台publish命令执行的结果，publish的topic需要设置为前面收到的命令请求的topic。平台如果在规定的时间内若没收到命令执行请求的响应，则会将此命令请求的状态设置为超时。若超时后才收到此命令执行的响应，依然会被丢弃。