**新大陆物联网云平台**

**设备接入MQTT协议**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **修订内容** | **创建/修订者** |
| v1.1 | 2017-07-19 | 初始版本 | cs |
| v2.0 | 2019-03-29 | 修改CONNECT的UserName值为项目ID；加入订阅/取消订阅功能；补充接入流程的详细说明等 | cs |
| v2.1 | 2019-06-13 | 修改命令请求与响应的cmdid为字符型 | cs |
| v2.2 | 2019-08-20 | 添加CONNECT的鉴权方式 | cs |

# 说明

MQTT逐渐成为物联网的重要组成部分。该协议支持所有平台，几乎可以把所有联网物品和外部连接起来，被用来当做传感器和执行器的通信协议。MQTT原版协议介绍请参见MQTT version 3.1.1官方文档，本文档对此不做详细说明。

# 接入流程

访问平台<http://www.nlecloud.com> 注册帐号

平台以项目为单位，一个项目下可以有多个设备，每个设备可以包含多个传感器，所以依次添加项目、设备、传感器，其中在添加设备时在“**通讯协议**”一栏中可以选择TCP\MQTT\HTTP等方式

操作以上功能时顺便记录下：设备标识(device)、设备传输密钥(key)等信息

硬件设备发送TCP连接请求到以下地址，发送以下各项数据报文与平台进行交互

平台接入服务器地址newgateway.nlecloud.com或IP 120.77.58.34，MQTT端口1883

# MQTT Packet格式说明

包格式包含三部分：

|  |  |
| --- | --- |
| Fixed Header | 所有packet中都必须有 |
| Varable Header | 部分包含有 |
| Payload | 部分包含有 |

## 3.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 - 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

该版本支持的所有类型：

|  |  |  |  |
| --- | --- | --- | --- |
| 名字 | 值 | 流向 | 描述 |
| CONNECT | 1 | C->S | 客户端请求与服务端建立连接 |
| CONNACK | 2 | S->C | 服务端确认连接建立 |
| PUBLISH | 3 | C⬄S | 发布消息 |
| PUBACK | 4 | C⬄S | 收到发布消息确认 |
| PUBREC | 5 | C⬄S | 发布消息收到 |
| PUBREL | 6 | C⬄S | 发布消息释放 |
| PUBCOMP | 7 | C⬄S | 发布消息完成 |
| SUBSCRIBE | 8 | C->S | 订阅请求 |
| SUBACK | 9 | S->C | 订阅确认 |
| UNSUBSCRIBE | 10 | C->S | 取消订阅 |
| UNSUBACK | 11 | S->C | 取消订阅确认 |
| PING | 12 | C->S | 客户端发送PING(连接保活)命令 |
| PINGRSP | 13 | S->C | PING命令回复 |
| DISCONNECT | 14 | C->S | 断开连接 |

## 3.2 Variable Header &Payload

|  |  |  |
| --- | --- | --- |
| 消息类型 | Variable Header | Payload |
| CONNECT | 有 | 有 |
| CONNACK | 有 | 有 |
| PUBLISH | 有 | 有 |
| PUBACK | 有 | 无 |
| PUBREC | 有 | 无 |
| PUBREL | 有 | 无 |
| PUBCOMP | 有 | 无 |
| SUBSCRIBE | 有 | 有 |
| SUBACK | 有 | 有 |
| UNSUBSCRIBE | 有 | 有 |
| UNSUBACK | 有 | 无 |
| PING | 无 | 无 |
| PINGRSP | 无 | 无 |
| DISCONNECT | 无 | 无 |

# 支持的Packet列表

## CONNECT：连接请求(client)->(server)

客户端到服务端的网络连接建立后，客户端发送给服务端的第一个报文必须是CONNECT报文，请求报文格式如下：

### 4.1.1 Fixed Header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 - 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.1.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1-2 | ProtocolName Length | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| byte 3 | ‘M’ | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| byte 4 | ‘Q’ | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| byte 5 | ‘T’ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| byte 6 | ‘T’ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| Byte7 | Protocol Level | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Byte8 | Connect Flag | User  flag | Password  flag | WillRetain  Flag | WillQos Flag | | WillFlag | CleanSession  Flag | Reserve |
| Byte9-10 | KeepAlive |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**详细分解说明**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **名称** | **说明** | **示例** |
| Protocol Level | 版本 | 必须设置为4，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 |  |
| user flag  password flag | 用户名  密码 | 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/  Willretainflag/  WillQosFlag |  | **Will flag**为0时，**WillQosFlag**和**WillRetainFlag**必须为0，  **Will Flag**为1时，**WillQosFlag**只能为0、1、2。  **注：**  **1．**当该设备下一次正常上线后，服务器会清除之前的will 消息（包括retain消息）。  **2．**如果设备上线时设置了Will Topic和msg，且将WillRetainFlag设置为1，只有设备异常断开后，服务器才会将该Will msg分发给新的Will Topic的订阅者，**且Qos级别以订阅的Request qos级别为准。**  **3．**如果设备上线时设置了Will Topic和msg，且将WillRetainFlag设置为0，服务器不会存储该消息，当设备异常断开后会将该Will msg以设备设置的WillQosFlag的Qos级别进行分发 |  |
| Clean  Session Flag |  | 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，session需保持的内容包含：   * 客户端订阅的topic列表。 * 还未完成确认的Qos1、Qos2级别的publish消息   客户端保存session的内容包含：   1. 已经发送到服务端的但还没有收到确认的Qos1、Qos2消息列表。 2. 待发送的Qos0列表。   若客户端将clean session标志位设置为1，当其断开后，平台会清除设备的订阅列表及未完成确认的Qos1、Qos2的publish消息 |  |
| Reserve |  | 保留位，置0 |  |
| **KeepAlive** | 保活时间 | 每个客户端可自定义设置连接保持时间，最短60秒，最长65535秒 |  |

### 4.1.3 Payload

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bit** | **Description** | **是否必须** | **对应字段** | **格式** |
| Field1 | Client Identifier | 是 | device | 2字节字串长度 + utf8字串 |
| Field2 | UserName | 是 | projectId | 2字节字串长度 + utf8字串 |
| Field3 | Password | 是 |  | 2字节字串长度 + utf8字串 |

**详细分解说明**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **名称** | **说明** | **示例** |
| device | 设备标识 | 在平台上添加设备时的设备标识：   1. 新大陆网关：进入网关设置-》【参数设置】-》【系统参数】中的序列号 2. 新大陆农业网关：浏览器登录农业网关设置页面-》【设备状态】中的设备编号 3. 新大陆家居网关：进入平板的家居网关主界面，界面左上角的一行序列号 4. **其它的MCU/SOC/网关/手机等设备：可自行输入一个唯一的标识用于与平台连接** | PF12345678 |
| projectId | 项目ID | 在平台上设备所对应的上级项目ID | 6617 |

其中Password支持鉴权方式如下：

**方式1：普通鉴权方式**

Password直接设置为平台添加设备时生成的传输密钥SecretKey值（在平台上添加设备时自动生成的一串32长度的字符串，该值在全局内唯一），示例如下：

Password=” a5528869647040b4a88bb61dcb566eee”

**方式2：安全鉴权方式**

设备接入时需要使用 SecretKey 计算出token 来进行访问安全认证，   
本例中，Password= 经过SecretKey计算的token = clientId & username & method & timestamp & sign

* timestamp：表示访问过期时间戳 timestamp，当访问参数中的timestamp时间小于当前时间时，平台会认为访问参数过期从而拒绝该访问。
* method：表示签名算法类型。支持md5，sha1和sha256。
* sign：表示clientId,projectId,method,timestamp,secretKey五个参数值按固定排序依次拼接（参数间不需要任何拼接字符），使用method对应的签名算法加签。示例如下：

sign=sha256("mydevice50023sha2561566276142807a5528869647040b4a88bb61dcb566eee) token="mydevice&50023&sha256&1566276142807&" + sign

## CONNACK：连接响应(server)->(client)

硬件设备客户端发送连接请求后，服务端会发送响应消息，响应报文格式如下:

### 4.2.1 Fixed Header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 - 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.2.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | Acknowledge Flags | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Sp |
| byte 2 | Return Code | x | x | x | x | x | x | x | x |

**详细分解说明**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **名称** | **说明** | **示例** |
| Sp  (Session Present Flag) | 连接确认标志 | session信息在服务器已保持，置1；未保存，置0 | 1 |
| Return Code | 连接返回码 | 0：握手连接成功  1：握手连接失败-协议版本错误  2：握手连接失败-设备ID长度超限  3：握手连接失败-未添加设备  4：握手连接失败-设备鉴权失败  5：握手连接失败-设备未授权连接  **失败：**  \*如果connect包不符合协议内容约束，则直接断掉连接，而不需要发送connack包.  \*如果鉴权或授权失败，回复一个带非0错误码的connack包.  **成功：**  \*必须断掉重复的clientid.  \*执行cleansession 对应的操作.  \*必须回复一个connack，回复码为0.  \*开始消息传递，并加入keepalive的监视 | 0 |

## PUBLISH：数据上报(client)->(server)

设备与服务器建立连接后，便可以进行传感数据的上报上传，上报报文格式如下：

### 4.3.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | DUP flag | QoS Level | | RETAIN |
| byte2 - 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

**详细分解说明**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **名称** | **说明** | **示例** |
| DUP flag | 重发标志 | 目前不支持该逻辑 | 1 |
| QoS Level | 服务质量等级 | |  |  |  |  | | --- | --- | --- | --- | | Bit2 | Bit1 | 值 | 描述 | | 0 | 0 | 0 | 最多发送一次 | | 0 | 1 | 1 | 至少发送一次 | | 1 | 0 | 2 | 只发送一次 | | 0 |
| RETAIN | 保留标志位 | 目前不支持该逻辑 |  |

### 4.3.2 VariableHeader

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Description** | **格式** | **是否必须** |
| Field1 | TopicName | 2字节字串长度 + utf8字串 | 是 |
| Field2 | PacketIdentifier | 2字节 | QoS0:否，QoS1、Qos2:是 |

**详细分解说明**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **名称** | **说明** | **示例** |
| TopicName | 主题名 | 固定为“**/sensor/datas**”，为上报传感数据的指令 | $datas |
| PacketIdentifier | 报文标识码 | QoS0:否，QoS1、Qos2:是 |  |

### 4.3.3 Payload

要上报的传感数据数组，支持的格式如下

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON键** | **JSON值** | **说明** | **报文示例** |
| datatype | 数据上报格式类型 | 具体为datas属性内的 传感数据格式类型，如下  = 1：JSON格式1字符串；  = 2：JSON格式2字符串；  = 3：JSON格式3字符串； | 1 |
| datas | 要上报的传感数据数组 | 该属性根据datatype类型的不同，可以上报多个传感器数据，也可以上报同一传感器的多条数据，其中**apitag1**为传感的标识名，**value**为传感值，可以是数字、浮点、字符串、二进制（最大48字节大小）  *数据类型为1（JSON格式1字符串）：*  **"**datas**"**:  {  **"**apitag1**"**: **"**value1**"**,  **"**apitag2**"**: value2,  …  } | 示例：  **"**datas**"**:  {  **"**temperature**"**: 23.5,  **"**rgb-r**"**: **"**#999**"**,  …  } |
| *数据类型为2（JSON格式2字符串）：*  apitag1与value数据格式同上，datetime1须是yyyy-mm-dd hh:mm:ss格式  **"**datas**"**:  {  **"**apitag1**"**:{**"**datetime1**"**:**"**value1**"**},  **"**apitag2**"**: {**"**datetime2":"value2"},  …  } | 示例：  "datas":  {  "temperature": {"2015-03-22 22:31:12”:22.5},  …  } |
| *数据类型为3（JSON格式3字符串）示例：*  value数据格式同上，  dt须是yyyy-mm-dd hh:mm:ss格式  "datas":  [  {  "apitag":"temperature",  "datapoints":  [  {  "dt":"2018-01-22 22:22:22", //可选  "value": 36.5 //数字浮点字符串  }  ]  },  {  "apitag": "location",  "datapoints":  […]  },  { … }  ] |  |
| msgid | 消息编号 | 由客户端生成的一个用于表示该条报文的编号，用于服务器下发“上报响应”时原样带回 | 123 |

**关于数据上报后的“服务端响应”机制注意事项：**

1. **Qos0时：服务端不发送任何响应**
2. **Qos1时：服务端给客户端回复相应的PubAck (报文格式参考4.5)**
3. **Qos3时：服务端给客户端回复相应的PubRec(报文格式参考4.5)；客户端再回复服务端PubRel报文(报文格式参考4.6)，如超时不回平台则会断开相应连接；反之服务端最后给客户端回复PubComp报文(报文格式参考4.7)**

## PUBLISH：命令请求(server)->(client)

设备成功连接云平台后，除了进行传感数据的上报，服务端也能下发命令，如控制某个传感器的开关，报文格式如下（使用publish报文来下发命令）：

### 4.4.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | DUP flag | QoS Level | | RETAIN |
| byte2 - 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

**详细分解说明**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **名称** | **说明** | **示例** |
| DUP flag | 重发标志 | 设为固定值0 | 0 |
| QoS Level | 服务质量等级 | 命令请求以Qos0级别进行下发 | 0 |
| RETAIN | 保留标志位 | ~~如果服务器分发的消息是RETAIN消息，则将该标志位置1，否则置0~~ |  |

### 4.4.2 VariableHeader

|  |  |  |
| --- | --- | --- |
|  | **Description** | **格式** |
| Field1 | TopicName | utf8字串 |

**详细分解说明**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **名称** | **说明** | **示例** |
| TopicName | 主题名 | 固定为“**/sensor/cmdreq**” |  |

### 4.4.3 Payload

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON键** | **JSON值** | **说明** | **消息示例** |
| cmdid | 命令编号 | 由服务端生成一个报文的字符串编号，客户端设备收到命令处理后上发“命令响应”时原样带回服务端 | “123456789” |
| apitag | 传感标识名(可空) | 为在平台上添加传感器时的标识名 | rgb\_open |
| data | 命令值 | 一个命令值，可以是数字、浮点、字符串、JSON | 例  数字：1  浮点：12.3  字符串：“你好”  JSON：{“onoff”:1 , “red” : 23.5} |

## PUBLISH：命令响应(client)->(server)

该报文由客户端发往服务器，以响应相应的命令，报文格式如下（使用publish报文来响应命令）：

### 4.5.1 Fixed header

参考4.3.1

### 4.5.2 VariableHeader

|  |  |  |
| --- | --- | --- |
|  | **Description** | **格式** |
| Field1 | TopicName | utf8字串 |

**详细分解说明**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **名称** | **说明** | **示例** |
| TopicName | 主题名 | 固定为“**/sensor/cmdresp**” |  |

### 4.5.3 Payload

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON键** | **JSON值** | **说明** | **消息示例** |
| cmdid | 命令编号 | 命令请求时的字符串命令编号，原样还回 | “123456789” |
| status | 状态结果 | 一个字节表示  0：处理成功；  1：处理失败；  其它：保留值； | 0 |
| data | 命令响应值 | 响应值，可以是数字、浮点、字符串、JSON | 同上 |

**关于命令响应后的“服务端响应”机制注意事项：**

1. **Qos0时：服务端不发送任何响应**
2. **Qos1时：服务端给客户端回复相应的PubAck (报文格式参考4.5)**
3. **Qos3时：服务端给客户端回复PubRec(报文格式参考4.5)；客户端再回复服务端PubRel报文(报文格式参考4.6)，服务端最后给客户端回复PubComp报文(报文格式参考4.7)**

## PUBACK

### 4.6.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.6.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | PacketIdentifier | PacketIdentifier | | | | | | | |

## PUBREC

### 4.7.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.7.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | PacketIdentifier | PacketIdentifier | | | | | | | |

## PUBREL

### 4.8.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.8.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | PacketIdentifier | PacketIdentifier | | | | | | | |

## PUBCOMP

### 4.9.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.9.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | PacketIdentifier | PacketIdentifier | | | | | | | |

## SUBSCRIBE

### 4.10.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.10.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | PacketIdentifier | PacketIdentifier | | | | | | | |

### 4.10.3 Payload

|  |  |  |  |
| --- | --- | --- | --- |
|  | Description | **说明** | **格式** |
| Byte1~n | TopicName | 主题名 | 2字节字串长度 + utf8字串  可以包含一个或多个topic，topic必须是数字、英文、反斜杠（/）的组合，目前不支持通配符。  以下topic被系统保留使用（客户端不能订阅）：  /sensor开头 |
| Byten+1 | Request Qos | 质量等级 | 服务质量要求(只能为0、1、2) |

## SUBACK

### 4.11.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.11.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | PacketIdentifier | PacketIdentifier | | | | | | | |

### 4.11.3 Payload

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | retcode |  | | | | | | | |

**返回码说明：**

|  |  |
| --- | --- |
| 返回码 | 描述 |
| 0x00 | 成功(granted qos = 0) |
| 0x01 | 成功(granted qos = 1) |
| 0x02 | 成功(granted qos = 2) |
| 0x80 | 失败 |

## UNSUBSCRIBE

### 4.12.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.12.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | PacketIdentifier | PacketIdentifier | | | | | | | |

### 4.12.3 Payload

|  |  |  |  |
| --- | --- | --- | --- |
|  | Description | **说明** | **格式** |
| Byte1~n | TopicName | 主题名 | 2字节字串长度 + utf8字串  可以包含一个或多个topic |

## UNSUBACK

### 4.13.1 Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

### 4.13.2 VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | PacketIdentifier | PacketIdentifier | | | | | | | |

## PING: 心跳请求(client)->(server)

MQTT的心跳请求是客户端主动向服务端发送，是通知服务端自己状态的一种机制，如果双方在本文CONNECT协议部分设置KeepAlive超时时间内没有任何报文交互（包括此心跳报文），则服务端将主动关闭该连接。固定报文如下：

## Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

## VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | NULL | NULL | | | | | | | |

注：PING命令无VariableHeader

## PINGRSP: 心跳响应(server)->(client)



## Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

## VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | NULL | NULL | | | | | | | |

注：PINGRSP无VariableHeader

## DISCONNECT：断开连接(server)->(client)



## Fixed header

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1 | MQTT Packet Type | | | | 0 | 0 | 0 | 0 |
| byte2 ~ 5 | Remaining Length（该字段占用1-4个字节） | | | | | | | |

## VariableHeader

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Description** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| byte 1~2 | NULL | NULL | | | | | | | |

注：DISCONNECT无VariableHeader

# 流程详解

## 设备连接



1. 设备向平台发起CONNECT请求，CONNECT中携带握手信息，具体参见（报文格式参考4.1）；
2. 平台拿到握手信息进行鉴权；
3. 握手通过后，如果CleanSession=0, 平台将会加载上次保存的设备一些信息（如果有保存），如订阅列表(4.1.2中描述)；

如果CleanSession=1, 平台将不加载这些信息（不管上次有没保存）；

1. 返回握手结果CONNACK（报文格式参考4.2）。

## 消息发布

## 数据上报

## Qos0(Client -> Server)



1. 设备使用PUBLISH Packet格式发布Qos0消息，具体参见（报文格式参考4.3）
2. 平台收到上报传感数据后保存起来

## Qos1(Client -> Server)



1. 设备使用PUBLISH Packet格式发布Qos1消息，具体参见（报文格式参考4.3）
2. 平台收到上报传感数据后保存起来
3. 平台给设备回复相应的PubAck (报文格式参考4.6)

## Qos2(Client -> Server)



1. 设备使用PUBLISH Packet格式发布Qos2消息，具体参见（报文格式参考4.3）
2. 平台收到上报传感数据后保存起来
3. 平台给设备回复相应的PubRec报文
4. 设备需回复平台PubRel报文，如超时不回平台则会断开相应连接
5. 平台给设备回复PubComp报文

**注：数据上报功能不支持Retain特性**

## 命令控制

## 命令下发

## Qos0(Server -> Client)



1. 平台使用PUBLISH Packet格式发布topic为/sensor/cmdreq的Qos0消息
2. 设备收到消息后，将做为控制执行器的指令来处理控制（报文格式参考4.5）。

**注：目前命令控制只以Qos0级别进行下发**

## 命令回复

## Qos0(Client -> Server)



1. 设备使用PUBLISH Packet格式发布Qos0消息
2. 平台收到响应后更新命令状态(报文格式参考4.5)

## Qos1(Client <-> Server)



1. 设备使用PUBLISH Packet格式发布Qos0消息
2. 平台收到响应后向设备回复一个Puback报文(报文格式参考4.5)，同时更新命令状态。

## Qos2(Client <-> Server)



1. 设备使用PUBLISH Packet格式发布Qos0消息
2. 平台收到响应后向设备回复一个PubRec报文
3. 设备在收到PubRec后需向平台回复PubRel报文
4. 平台收到PubRel报文后，向设备回复PubComp报文，同时更新命令状态。

## 订阅



1. 设备发起订阅请求(报文格式参考4.10)
2. 平台收到请求后更新topic列表.
3. 平台给设备回复SubAck(报文格式参考4.11)

**注：subscribe的request qos级别可以为0、1、2**

## 取消订阅



1. 设备发起取消订阅请求(报文格式参考4.12)
2. 平台收到请求后更新topic列表.
3. 平台给设备回复UnSubAck(报文格式参考4.13)

## 推送设备

## Publish Packet推送

**Qos0：设备使用Qos级别为0的Publish报文来进行推送**



1. 设备发起推送topic请求(以Qos0级别)
2. 平台收到请求后,将topic以Qos0级别推送到相关订阅设备

**Qos1：设备使用Qos级别为1的Publish报文来推送Topic**



1. 设备发起推送topic请求(以Qos1级别)
2. 平台收到请求后，回复设备PubAck报文，同时将topic以Qos1级别推送到相关订阅设备
3. 订阅设备收到推送信息后返回PubAck报文给平台

**Qos2：设备使用Qos级别为2的Publish报文来推送Topic**



1. 设备发起推送topic请求(以Qos2级别).
2. 平台收到请求后,回复PubRec报文
3. 设备收到PubRec后需回复PubRel报文平台收到PubRel报文后,回复PubComp给设备。
4. 平台在回复PubComp后会以Qos2级别推送到相关订阅设备
5. 设备需回复PubRec报文平台发送PubRel报文给设备
6. 设备需回复PubComp(发布完成)