**MQTT详细设计**

**北京恒泰博远科技有限公司**

**2018-10-30**

# 

**更新记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **描述** | **作者** |
| 2018年10月30日 | 1.0 | 初始化 | 周西栋 |
| 2018年11月09日 | 1.1 | 添加日志主题，见2.2、2.7、4.1 | 周西栋 |
| 2018年11月12日 | 2.1 | 修改采集代理的配置协议 | 周西栋 |
| 2018年11月21日 | 2.2 | 修改了日志协议，添加应答子类型 | 周西栋 |
|  |  |  |  |
|  |  |  |  |

**目 录**

[1. MQTT设计 4](#_Toc530579415)

[1.1. broker部署位置 4](#_Toc530579416)

[1.2. 交互模式 4](#_Toc530579417)

[1.3. 消息定位 4](#_Toc530579418)

[1.4. 特性介绍 5](#_Toc530579419)

[2. TOPIC设计 5](#_Toc530579420)

[2.1. 服务类型 5](#_Toc530579421)

[2.2. 功能类型 5](#_Toc530579422)

[2.3. 服务发现 5](#_Toc530579423)

[2.4. 心跳检测 6](#_Toc530579424)

[2.5. 订阅主题 6](#_Toc530579425)

[2.6. 数据采集 7](#_Toc530579426)

[2.7. 日志主题 7](#_Toc530579427)

[3. 数据类型 8](#_Toc530579428)

[3.1. 类型 8](#_Toc530579429)

[3.2. 流程 8](#_Toc530579430)

[4. Message协议 8](#_Toc530579431)

[4.1. msg协议 8](#_Toc530579432)

[4.2. body协议 9](#_Toc530579433)

[5. 采集数据协议 10](#_Toc530579434)

[5.1. 数据消息 10](#_Toc530579435)

[5.2. 数据消息回复 10](#_Toc530579436)

[6. 控制协议 10](#_Toc530579437)

[6.1. 控制消息 10](#_Toc530579438)

[6.2. 控制消息回复 10](#_Toc530579439)

[7. 配置协议 10](#_Toc530579440)

[7.1. 配置消息 10](#_Toc530579441)

[7.2. 采集代理配置 10](#_Toc530579442)

[7.3. 物联网接口服务器采集服务配置 14](#_Toc530579443)

[7.4. 物联网接口服务器控制服务配置 14](#_Toc530579444)

[7.5. 配置消息回复 15](#_Toc530579445)

[8. 查询配置信息 15](#_Toc530579446)

[8.1. 状态协议 15](#_Toc530579447)

[8.2. 状态消息 15](#_Toc530579448)

[8.3. 状态消息回复 16](#_Toc530579449)

# MQTT设计

## broker部署位置

broker是mqtt的一个代理服务，目前系统中设计了两类broker：采集代理与物联网接口之间的broker1和物联网接口与除采集代理之外的服务之间的broker2。

* broker1：交互采集代理与物联网接口之间的通讯信息。
* broker2：交互除采集代理以外的各服务之间的通讯信息。

## 交互模式

mqtt使用发布和订阅模式：

* 发布：向broker发送数据消息。
* 订阅：从broker读取数据消息。

## 消息定位

mqtt 利用topic（主题）对消息区分：

* topic可以用“/”分层。
* mqtt 提供了对topic的便捷管理，如“#”通配符和“+”占位符。如果broker上有a/b/c这样的分层topic，订阅时可用“a/#”和“a/+/c”对其订阅。

**注意**：

 不要在最前面加 /。比如: /home/device/light, 等于在最前面有一个空字符串层级，这完全没有必要而且增加了broker之类的处理, home/device/light才是合理的。

 就使用英文+数字字符，不要使用空格、特殊字符，这些都会增加处理的复杂性，有时候还会有兼容性问题。

 将设备ID或识别码包含在topic中。多方面原因：便于订阅特定设备、便于后续的过滤、便于权限控制

 不要在运行中创建 topic。这个好理解，交流还是有预期比较好。在规模较大的系统中，随意的创建 topic 会导致维护困难、处理复杂度增加，最主要的肯定非常容易导致遗漏和未知行为。

 命名简单明确。太长了看太累，看不懂会困惑，太累或困惑都容易出错； topic 应该尽可能细致，能详细定位到不同的设备和消息

 命令字保持一致，并放在最后。MQTT不像HTTP有多个方法，RESTful因此将CRUD映射为HTTP方法。当然MQTT通常只有一个控制下发命令字，状态和属性是主动上报的。MQTT可以将方法放在topic最后，这样也便于统一的下发处理， 比如: home/bedroom/bedlight/rgb/set

## 特性介绍

mqtt 严格意义上来讲，不是消息队列，而是一个类似于redis的缓存服务。以此来看，对某一个topic发送了多条数据时（a ,b ,c ,d ），如果没有先前订阅这个topic的客户端，则在broker上topic存在的数据为最后接收到的数据 ( d )，之前的数据全被覆盖了( a , b , c 不存在了)。如果没有发布某个主题topic，则在订阅次主题topic时，broker上不会主动创建topic；而在发布时如果不存在topic，则broker会创建相应的topic。

# TOPIC设计

## 服务类型

|  |  |  |
| --- | --- | --- |
| 服务名称 | 服务类型 | 所在broker |
| 采集代理 | PROXY | broker1 |
| 物联网接口 | COLLECT | broker1、broker2 |
| Web服务 | WEB | broker2 |
| 控制接口服务 | CONTROL | broker2 |
| 分发接口服务 | DISTRIBUTE | broker2 |

## 功能类型

|  |  |  |
| --- | --- | --- |
| 功能名称 | 功能类型 | 说明 |
| 注册信息 | REGIST | 只能用在服务发现（见2.3），存放服务信息注册 |
| 心跳信息 | PING | 只能用在心跳检测（见2.4），存放各服务的心跳信息 |
| 配置信息 | CONFIG | 只能用在业务功能（见2.5），存放服务的配置信息 |
| 数据信息 | DATA | 只能用在业务功能（见2.5），存放采集上来的数据信息 |
| 状态信息 | STATUS | 只能用在业务功能（见2.5），存放各服务的状态信息 |
| 应答信息 | ANSWER | 适用于所有，存放应答信息 |
| 控制信息 | CTRL | 适用于所有，存放控制信息 |
| 日志信息 | LOG | 适用于所有，存放日志信息 |

## 服务发现

* 注册服务信息：

各服务启动时将自身的服务信息向特定的主题REGIST/serverType/macAddress中发布，而web和物联网接口服务则会从相应的主题中将注册信息读取出来，完成入库等动作。

* 读取服务消息：

Web服务从broker2上的REGIST/serverType/macAddress上读取服务的配置信息，如果读到了消息则表明该服务存在，然后将收到的消息注册（入库）。

topic各部分含义注释：

 REGIST ：是特定字段，为了方便订阅者统一订阅注册信息使用。

 serverType ：是指注册者的服务类型， 参考2.1。

 macAddress：注册者的mac地址，如果存在多个则选用其中一个值较大者，地址内容不应携带符号，且字母为大写，例如：5c:96:9d:71:f5:a3应取5C969D71F5A3。

## 心跳检测

topic格式 ：PING/serverType/macAddress

topic各部分含义说明：

 PING ：是特定字段，为了方便订阅着统一订阅心跳信息使用。

 serverType ：是指注册者的服务类型， 参考2.1。

 macAddress：注册者的mac地址，如果存在多个则选用其中一个值较大者，地址内容不应携带符号，且字母为大写，例如：5c:96:9d:71:f5:a3应取5C969D71F5A3。

## 订阅主题

* 主题发布

当服务发布自身的注册信息后，要发布自身需要订阅的主题。因为在订阅主题时，如果主题不存在，mqtt会中断连接，触发重连机制，而长时间订阅不到主题，会影响其他已有的订阅功能。所以，需要在订阅之前，先用发布空消息的方式创建所需的订阅主题。

* 主题格式

订阅主题格式统一为：fromServerType/toServerType/toMacAddress/functionType

 fromServerType ：消息发布者的服务类型，参考2.1。

 toServerType ：消息订阅者的服务类型，参考2.1。

 toMacAddress：消息订阅者的mac地址，如果存在多个则选用其中一个值较大者，地址内容不应携带符号，且字母为大写，例如：5c:96:9d:71:f5:a3应取5C969D71F5A3。

 functionType ：除注册和心跳外的功能类型，参考2.2。

## 数据采集

* 发布消息

topic格式： fromServerType/toServerType/ fromMacAddress /toMacAddress/functionType

topic各部分含义说明：

 fromServerType ：消息发布者的服务类型，参考2.1。

 toServerType ：消息订阅者的服务类型，参考2.1。

 fromMacAddress：消息发布者的mac地址，如果存在多个则选用其中一个值较大者，地址内容不应携带符号，且字母为大写，例如：5c:96:9d:71:f5:a3应取5C969D71F5A3。

 toMacAddress：消息订阅者的mac地址，如果存在多个则选用其中一个值较大者，地址内容不应携带符号，且字母为大写，例如：5c:96:9d:71:f5:a3应取5C969D71F5A3。

 functionType ：除注册和心跳外的功能类型，参考2.2。

* 订阅消息

订阅消息时使用的topic的规则就是发布消息时使用的topic。

## 日志主题

topic格式 ：LOG/serverType/macAddress(web用)

fromServerType /toServerType/toMacAddress/LOG(物联网接口服务用)

topic各部分含义说明：

 PING：是特定字段，为了方便订阅着统一订阅心跳信息使用。

 serverType ：是指注册者的服务类型， 参考2.1。

 macAddress：注册者的mac地址，如果存在多个则选用其中一个值较大者，地址内容不应携带符号，且字母为大写，例如：5c:96:9d:71:f5:a3应取5C969D71F5A3。

 fromServerType ：消息发布者的服务类型，参考2.1。

 toServerType ：消息订阅者的服务类型，参考2.1。

 toMacAddress：消息发布者的mac地址，如果存在多个则选用其中一个值较大者，地址内容不应携带符号，且字母为大写，例如：5c:96:9d:71:f5:a3应取5C969D71F5A3。

# 数据类型

## 类型

在数据传输时，传递的是byte[]（java中还有一种发布方式，就是发布org.eclipse.paho.client.mqttv3.MqttMessage对象，但是最终也是将要发的数据转成byte[]，并存放在了MqttMessage的payload属性中；消息质量在MqttMessage对象中定义，对应的属性是qos。）

## 流程

消息从发布到被订阅到，整个流程数据的变化如下：

对象 > json > byte[] > 发布 > broker > 订阅 > byte[] > json > 对象

# Message协议

## msg协议

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 说明 | 类型 | 备注 |
| msg\_id | 消息id | String | 格式为：发布者mac地址\_时间戳 |
| callback\_id | 应答的目标 消息id | String | 格式为：应答的目标mac地址\_时间戳 |
| msg\_type | msg类型 | Integer | 1 注册消息  2 心跳消息  3 业务消息 4 配置消息 5 服务状态消息 6 应答消息 7 日志消息 |
| source\_mac | 发布者的mac地址 | String | 发布者所在服务器的mac地址 |
| source \_type | 发布者类服务类型 | String | 发布者所属的服务类型，见2.1说明。 |
| create\_time | 创建时间 | Long | 时间戳 |
| license | 节点授权码 | String | 由服务器端发起授权 |
| body | 协议体 | String | Json格式，具体格式参见4.2。 |

## body协议

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | | 类型 | 说明 | 备注 |
| 消息子类型 | | Integer | 消息子类型的分类 | 最好每个协议里只放同一类消息 |
| 消息体 | 消息体1 | String | 具体消息 | 如果是对象，则转成json格式 |
| 消息体2 | String | 具体消息 | 同上 |
| 消息体3 | String | 具体消息 | 同上 |
| 消息体4 | String | 具体消息 | 同上 |
| .... | String | 具体消息 | 同上 |

Message协议体由2部分构成

1）消息子类型，标示业务处理子类型。

2）消息体，标示业务处理需要的对象实体，服务间通讯时用json格式。

补充说明：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能类型 | 用途 | 子类型 | 说明 |
| ANSWER | 采集代理 | 1 | 同步设备和变量，应答内容：“CONFIG” |
| 2 | 控制命令，应答内容：“CTRL” |
| 3 | 心跳，应答内容：“PING” |
| 4 | 服务状态，应答内容：“STATUS” |
| 5 | 日志，应答内容：“LOG” |
| 物联网接口 | 1 | 同步设备和变量，应答内容：“CONFIG” |
| 2 | 采集代理接收到配置，应答内容：“CONFIG” |
| 3 | 采集代理接收到控制，应答内容：“CTRL” |
| 4 | Web收到设备和变量，应答内容：“CONFIG” |
| 5 | 控制接口收到控制结果，应答内容：“CTRL” |
| 控制接口 | 1 | 物联网接口收到控制，应答内容：“CTRL” |
| 分发接口 |  |  |
| web管理端 | 1 | 物联网接口收到配置的应答 |
| 2 | 控制接口收到配置的应答 |
| 3 | 分发接口收到配置的应答 |
| CONFIG | 采集代理～物联网接口 | 1 | 设备和变量的信息 |
| 2 | 采集代理的配置信息 |
| 物联网接口～web | 1 | 物联网接口的配置信息 |
| 2 | 设备和变量的信息 |
| 3 | 采集代理的配置信息 |
| 控制接口～web | 1 | 控制接口的配置信息 |
| 分发接口～web | 1 | 分发接口的配置信息 |

# 采集数据协议

## 数据消息

[http://域名/IOServer标识/分组标识/设备标识/变量标识,](http://域名/IOServer标识/分组标识/设备标识/变量标识/)变量值,时间戳;

注意 ： 变量标示 变量值 时间戳 之间是用“,”分割开的。

## 数据消息回复

状态值（整型，含义参见状态值和错误码统一文档）;备注字符串

# 控制协议

## 控制消息

<http://域名/IOServer标识/分组标识/设备标识/变量标识/>写变量值\_时间戳\_超时时间\_尝试次数

## 控制消息回复

状态值（整型，含义参见状态值和错误码统一文档）;备注字符串

# 配置协议

## 配置消息

json字符串

配置消息较长，可以通过协议头sequence\_sub\_id来控制分包处理。

## 采集代理配置

文件格式：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | | | | 类型 | 备注 |
| id | |  | | Integer | 服务id |
| name | |  | | String | 服务名称 |
| address | |  | | String | 服务地址 |
| domain | |  | | String | 域名 |
| group\_id | |  | | Integer | 分组标示 |
| IOServer id | |  | | Integer | 服务id |
| IOServer address | |  | | String |  |
| IOServer name | |  | | String |  |
| device\_list | device | id | | Integer | 设备id |
| name | |  | 设备名称 |
| addr | | String | 设备地址 |
| time\_out | | Integer | 设备通讯超时时间（秒） |
| time\_span | | Integer | 默认打包间隔时间（秒） |
| device |  | |  |  |
| var\_list | var\_info | id | | Integer | 数据库变量id |
| var\_id | | Integer | IOServer上变量id |
| name | | String | 变量名称 |
| device\_id | | String | 关联设备ID |
| info | | String | 变量说明 |
| error\_as\_null | | Integer | 变量错误或者超时是否传输NULL |
| time\_out | | Integer | 变量超时时间（秒） |
| pack\_type | | Integer | 打包类型：0:间隔时间打包；1：按分钟；2：每15分钟；3：每小时；4每天 |
| time\_span | | Integer | 默认打包间隔时间（秒） |
| var\_info |  | |  |  |
| pack\_list | pack\_info | id | | Integer | 数据包id |
| name | | String | 数据包名称 |
| group\_id | | String | 分组标识 |
| error\_as\_null | | Integer | 变量错误或者超时是否传输NULL |
| time\_out | | Integer | 变量超时时间（秒） |
| pack\_type | | Integer | 打包类型：0:间隔时间打包；1：按分钟；2：每15分钟；3：每小时；4每天 |
| time\_span | | Integer | 默认打包间隔时间（秒） |
| var\_list | id | Integer | 数据库变量ID |
| id… |  |  |
|  |  | |  |  |

示例：

{

"\_comment": "配置信息",

"id": "10001",

"name": "节点名称",

"address": "192.168.1.101:50125",

"IOServer address": "192.168.1.101:8899",

"IOServer name": "我的IOServer",

"\_device\_list \_comment": "设备列表信息",

"device\_list": [

{

"\_device\_comment": "设备1信息",

"device": {

"id": "1001",

"name": "设备1",

"addr": "",

"time\_span": "50",

"time\_out": "120"

}

},

{

"\_device\_comment": "设备2信息",

"device": {

"id": "1002",

"name": "设备名称2",

"addr": "",

"time\_span": "50",

"time\_out ": "120"

}

}

],

"\_var\_list \_comment": "采集变量列表信息",

"var\_list": [

{

"\_var\_comment": "采集变量信息",

"var\_info": {

"id": "1001",

"name": "变量名称1",

"info": "",

"device\_id": "1001",

"error\_as\_null": "0",

"pack\_type": "0",

"time\_span": "50",

"time\_out": "120"

}

},

{

"\_var\_comment": "采集变量信息",

"var\_info": {

"id": "1002",

"name": "变量名称2",

"addr": "",

"device\_id": "1002",

"error\_as\_null ": "1",

"pack\_type": "1",

"time\_span": "50",

"time\_out ": "120"

}

}

],

"\_packet\_list \_comment": "采集数据包列表信息",

"packet\_list": [

{

"\_packet\_comment": "采集数据包信息",

"pack\_info": {

"id": "1001",

"name": "包名称1",

"error\_as\_null ": "0",

"time\_span": "50",

"time\_out": "120",

"var\_list": [

{

"id": "1001"

},

{

"id": "1002"

}

]

}

}

]

}

## 物联网接口服务器采集服务配置

文件格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | | 说明 | 类型 | 备注 |
| id | |  | Integer | 服务id |
| name | |  | String | 服务名称 |
| address | |  | String | 服务地址 |
| storage\_topic | |  | String | 存储队列服务标识 |
| ioserver\_list | ioserver | id | Integer | IOServer id |
| name | String | IOServer名称 |
| addr | String | IOServer地址 |
| proxy\_id | Integer | proxy id |
| proxy\_name | String | proxy名称 |
| proxy\_addr | String | proxy地址 |
| time\_out | Integer | IOServer与proxy通讯超时时间（秒） |
| storage\_servicestatus\_topic | String | 存储队列服务状态标识 |
| storage\_devicestatus\_topic | String | 存储队列设备状态标识 |
| storage\_msg\_topic | String | 存储队列采集数据标识 |
| ioserver… |  |  |  |

## 物联网接口服务器控制服务配置

文件格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | | 说明 | 类型 | 备注 |
| id | |  | Integer | 服务id |
| name | |  | String | 服务名称 |
| address | |  | String | 服务地址 |
| storage\_topic | |  | String | 存储队列服务标识 |
| ioserver\_list | ioserver | id | Integer | IOServer id |
| name | String | IOServer名称 |
| addr | String | IOServer地址 |
| proxy\_id | Integer | proxy id |
| proxy\_name | String | proxy名称 |
| proxy\_addr | String | proxy地址 |
| time\_out | Integer | IOServer与proxy通讯超时时间（秒） |
| storage\_servicestatus\_topic | String | 存储队列服务状态标识 |
| storage\_msg\_topic | String | 存储队列控制数据标识 |
| storage\_msgstatus\_topic | String | 存储队列控制执行状态标识 |
| ioserver… |  |  |  |

## 配置消息回复

状态值（整型，含义参见《04\_状态值和错误码统一文档》）;备注字符串

# 查询配置信息

todo

## 状态协议

## 状态消息

服务状态：

[http://](http://域名/IOServer标识/分组标识/设备标识/变量标识/)domain[/](http://域名/IOServer标识/分组标识/设备标识/变量标识/)service\_status/service\_id[/](http://域名/IOServer标识/分组标识/设备标识/变量标识/)stauts\_cpu\_storageused-storagetotal\_memused-memtotal\_time

设备状态：

[http://](http://域名/IOServer标识/分组标识/设备标识/变量标识/)domain[/](http://域名/IOServer标识/分组标识/设备标识/变量标识/)device\_status/[IOServer标识/分组标识/设备标识](http://域名/IOServer标识/分组标识/设备标识/变量标识/)/status\_lastonlinetime

## 状态消息回复

状态值（整型，含义参见状态值和错误码统一文档）;备注字符串