机智云平台标准接入协议

之设备与云端通讯

(v4.0.6)

修订历史

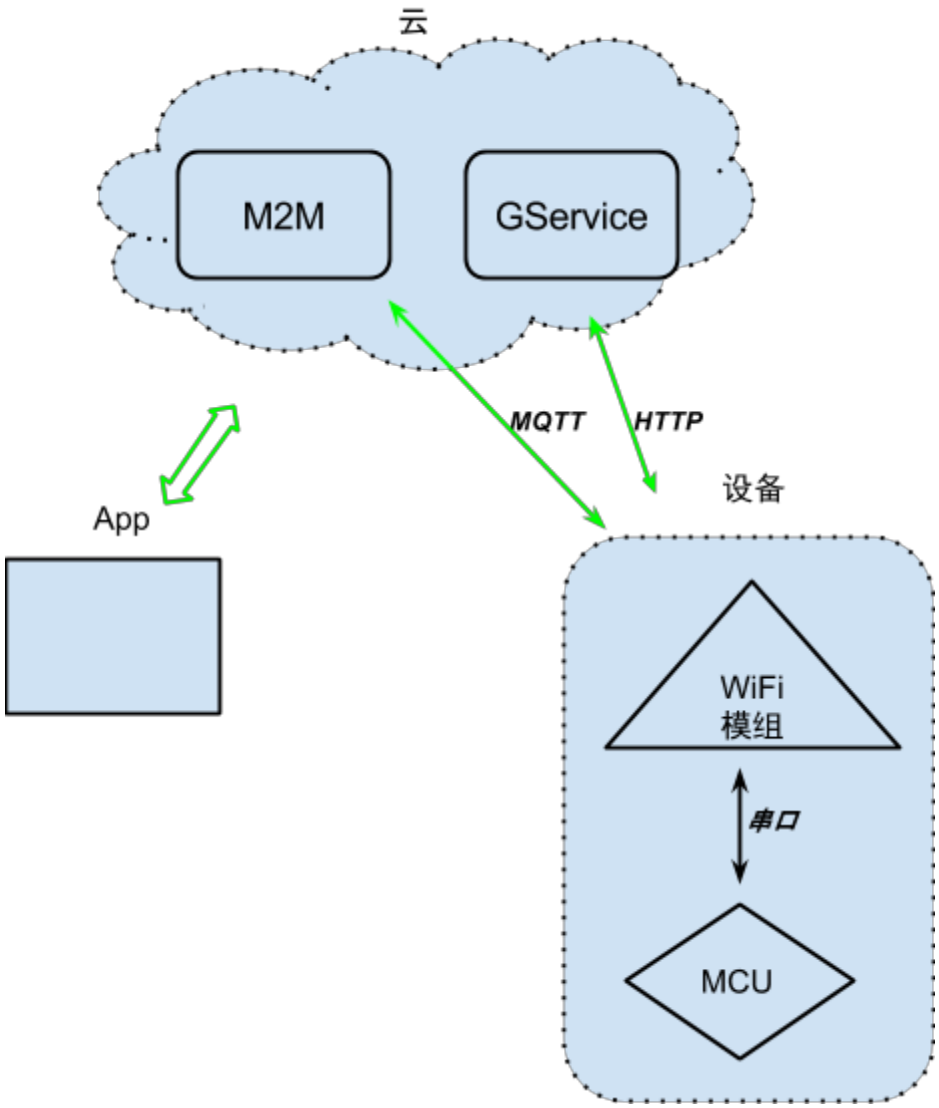
版本	修订内容	修订人	修订日期
4.0.0	V4版协议	刘冠华	2014-11-19
4.0.1	1. 添加“4.8 ” (p7) 2. 修正设备OTA中“设备收到固件升级通知”中的指令格式 (p4)	刘冠华	2014-11-29
4.0.2	1. 添加“4.8 设备请求在线App的数量” (p7) 2. 修改“4.9 通知设备在线App数量的变化” (p7)的命令字，取消定期推送	刘冠华	2014-12-05
4.0.3	更正文档错误：移除透传业务指令中的“业务指令长度”字段	刘冠华	2014-12-15
4.0.4	添加设备注销API(p3)	刘冠华	2014-12-17
4.0.5	1. 添加透传业务指令 (带ACK） (p7) 2. 添加设备OTA(v4.1) (p4)	刘冠华	2015-04-18
4.0.6	更新了“设备OTA (v4.1)”的描述文字，加入了对应的API文档的连接	刘冠华	2015-05-06

目录

设备向云端及App发送日志

1. 通讯模型

设备与云端通讯采用MQTT和HTTP两种方式。HTTP方式主要是提供设备注册，设备Provision，OTA升级等接口。而MQTT提供设备上报状态及设备接收远程控制指令的通道。MQTT是建立在TCP基础上的长连接通讯协议，详细请参考文档《MQTT_V3.1_Protocol_Specific.pdf》。



GService提供HTTP服务，而M2M提供MQTT实时通讯。

2. 通讯流程

设备与云端的大循环通讯主要包括以下的通讯过程。

- **设备注册。**设备首次上电时需要先向云端注册得到设备唯一标识码DID(Device ID)。

- **设备Provision。**设备每次连接M2M服务器前，都需先通过HTTP 获取M2M服务器的地址和端口号。
- **设备OTA。**设备每次启动后，向云端检查有没有新的固件可以更新。
- **设备上报状态和接受远程控制。**设备主动与M2M服务器建立长连接并保持该连接，并且当设备状态发生变化后或周期的主动向云端上报状态，同时会接受从该连接下发的远程控制指令。

3. 约定

3.1. 协议阅读说明

- **可变长度**：由 1～4 个字节(B)表示本可变长度字段后一直到数据包结尾的字节数。编码解码方式请参考《MQTT_V3.1_Protocol_Specific.pdf》第6页Remaining Length。
- **长度**：没有注明为可变长度的都为普通长度，一般由一个(1B)或两个字节(2B)组成。若多于一个字节组成，采用大端编码方式，即高字节在前，低字节在后。

3.2. 云端GService服务地址(HTTP)

- 服务器地址为：api.gizwits.com
- 服务端口号为：80

3.3. 设备与M2M服务器建立MQTT连接后，需定阅下以的主题以接收M2M服务器下发的消息。

- 接收服务器的消息响应：*ser2cli_res/<DID>* (<DID>指代设备的DID)
- 接收服务器的消息推送：*ser2cli_noti/<ProductKey>* (<ProductKey>指代设备产品标识码)
- 接收远程控制指令：*app2dev/<DID>/#* (<DID>指代设备的DID)

4. 通讯协议

4.1. 设备注册

新设备都必须先向云端注册后才能正常工作。

设备注册， 设备 ⇒ GService，HTTP。设备向云端提交设备标识码(ProductKey)，设备MAC，及自身生成的Passcode向云端注册，云端返回设备DID(Device ID)，设备保存好DID和Passcode完成注册过程。
请见http://docs.gizwits.apiary.io/reference/api-for-devices/devices/create-device。

4.2. 设备注销

设备被重置后都必须先向云端注销DID。

设备注销， 设备 ⇒ GService，HTTP。
请见http://docs.gizwits.apiary.io/#reference/api-for-devices/devices/disable-device。

4.3. 设备Provision

设备每次连接M2M服务器前，都需先获取M2M服务器的地址和端口号再连接对应的M2M服务器。

设备Provision，设备 ⇒ GService，HTTP。
请见http://docs.gizwits.apiary.io/reference/api-for-devices/device-provision/device-provision。

4.4. 设备OTA (v4.0)

设备每次启动后，向云端检查最新的固件ID，如最新的固件ID和本地保存的固件ID不一致，则进行远程固件升级。

设备向云端检查最新的固件ID，设备 ⇒ GService，HTTP。如发现返回的固件ID(target_fid)和本地保存的固件ID不一致，则通过返回的download_url进行固件下载并升级。
请见http://docs.gizwits.apiary.io/reference/api-for-devices/ota-get-target-fid/get-target-fid。

当设备正常在线运行时，云端可能也会下发固件升级通知到设备。设备比较目标固件ID决定是否升级与否。
设备收到固件升级通知，M2M服务器 ⇒ 设备，MQTT PUBLISH。主题为*ser2cli_res/<DID>*(<DID>指代设备的DID)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	命令字	2	0x020E
3	固件类型	1	1为WiFi固件，2为MCU固件
4	目标固件ID	4	目标固件ID，设备判断该固件ID和本地保存的固件ID是否一致进行固件升级
5	固件下载连接长度	2	len(固件下载连接)
6	固件下载连接	max 512	即download_url，通过这个连接可以下载固件

4.5. 设备OTA (v4.1)

设备每次启动后，向云端检查最新的固件SoftVersion，如最新的固件SoftVersion和本地保存的固件SoftVersion不一致，则进行远程固件升级。

设备向云端检查是否有新版固件需要升级，设备 ⇒ GService，HTTP。如发现返回的固件SoftVersion和本地保存的固件SoftVersion不一致，则通过返回的download_url进行固件下载并升级。请见

http://docs.gizwits.apiary.io/#reference/api-for-devices/ota-v41-update-firmware-info-and-check-latest/update-firmware-info-and-check-latest。

当设备正常在线运行时，云端可能也会下发固件升级通知到设备。设备由到通知后主动向云端检查是否有新版固件需要升级。
设备收到固件升级通知，M2M服务器 ⇒ 设备，MQTT PUBLISH。主题为*ser2cli_res/<DID>*(<DID>指代设备的DID)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	命令字	2	0x0211
3	固件类型	1	1为WiFi固件，2为MCU固件

4.6. 设备登陆云端(MQTT)

设备TCP连接上M2M服务器的服务端口后，需发送MQTT CONNECT消息登陆M2M服务器，M2M服务器地址和服务端口通过设备Provision获得。

设备登陆云端，设备 ⇒ M2M服务器，MQTT CONNECT。请求参数请根据下表填入。

MQTT参数	填写内容
Client Identifier	设备DID
User Name	设备DID
Password	设备Passcode
Keep Alive timer	最大值600秒，建议120秒。如超出最大值，服务器将按最大值处理

请见《MQTT_V3.1_Protocol_Specific.pdf》第15页3.1. CONNECT。

登陆响应，M2M服务器 ⇒ 设备，MQTT CONNACK。
请见《MQTT_V3.1_Protocol_Specific.pdf》第18页3.2. CONNACK。

4.7. 心跳

设备(以下简称客户端)与服务器建立连接后，需要定期向M2M服务器发送心跳，M2M服务器收到客户端的心跳请求后回复响应。

客户端向M2M服务器发送心跳请求，客户端 ⇒ M2M服务器，MQTT PINGREQ。
请见《MQTT_V3.1_Protocol_Specific.pdf》第33页3.12. PINGREQ。

M2M服务器回复心跳，M2M服务器 ⇒ 客户端，MQTT PINGRESP。M2M服务器回应心跳请求。
请见《MQTT_V3.1_Protocol_Specific.pdf》第34页3.13. PINGRESP。

客户端向M2M服务器发送心跳请求的最快频率为180秒内60次，心跳频率快于此值M2M服务器会主动断开与客户端的连接。建议客户端60秒发一次心跳。

客户端如连续2两次没有收到M2M服务器的心跳响应，可以认为与M2M服务器的连接已断开，尝试重连。

关于客户端与M2M服务器的重连策略。当客户端与服务器的连接异常断开后，客户端会重连服务器。第一次与服务器的连接断开后，马上重连，如连接不上，10秒后重连，如还是连接不上，20秒后重连，依此类推，每次重连失败后都延长10秒后再重连。如客户端重启后，重连间隔从头从0秒计起。

4.8. 透传业务指令

设备登陆M2M服务器后，App与设备相互间可以透过M2M服务器发送和接收业务指令，业务指令视具体的产品定制。

设备收到App透过M2M服务器传送的业务指令，M2M服务器 ⇒ 设备，MQTT PUBLISH。主题为*app2dev/<DID>/<AppClientId>* (<DID>指设备的DID，<AppClientId>指发出消息的App的当前MQTT Client Identifier，<AppClientId>在设备回复App时用到，见下文)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	可变长度	1~4	len(Flag...业务指令)
3	Flag	1	0x00
4	命令字	2	0x0090
5	业务指令	max 65535	业务指令

设备向App透过M2M服务器传送业务指令，设备 ⇒ M2M服务器，MQTT PUBLISH。主题为*dev2app/<DID>/<AppClientId>*，如设备想向所有与其有绑定关系的App传送业务指令，则主题为*dev2app/<DID>*。(<DID>指设备的DID，<AppClientId>指接收消息的App的当前MQTT Client Identifier，<AppClientId>是App向设备发送业务指令时在主题中带过来的，见上文)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	可变长度	1~4	len(Flag...业务指令)

3	Flag	1	0x00
4	命令字	2	0x0091
5	业务指令	max 65535	业务指令

4.9. 透传业务指令（带ACK）

设备登陆M2M服务器后，App与设备相互间可以透过M2M服务器发送和接收业务指令，业务指令视具体的产品定制。

发送需要ACK的业务指令，M2M服务器 ⇒ 设备 或 设备 ⇒ M2M服务器，MQTT PUBLISH。如是从App发往设备则主题为*app2dev/<DID>/<AppClientId>*，如是由设备发往App则主题为*dev2app/<DID>* 或 *dev2app/<DID>/<AppClientId>* (<DID>指设备的DID，<AppClientId>指发出消息的App的当前MQTT Client Identifier)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	可变长度	1~4	len(Flag...业务指令)
3	Flag	1	0x00
4	命令字	2	0x0093
5	包序号(sn)	4	消息的序号，用于消息的ACK
6	业务指令	max 65535	业务指令

消息ACK，设备 ⇒ M2M服务器 或 M2M服务器 ⇒ 设备，MQTT PUBLISH。如是从App发往设备则主题为*app2dev/<DID>/<AppClientId>*，如是由设备发往App则主题为*dev2app/<DID>* 或 *dev2app/<DID>/<AppClientId>* (<DID>指设备的DID，<AppClientId>指发出消息的App的当前MQTT Client Identifier)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	可变长度	1~4	len(Flag...业务指令)
3	Flag	1	0x00
4	命令字	2	0x0094
5	包序号(sn)	4	对应发送包的包序号
6	业务指令	max 65535	业务指令

注意：M2M服务器不会对消息进行ACK，消息的ACK由接收端负责，即为设备或App。

4.10. 设置设备的日志级别与指示灯开关

可以设置设备的日志级别与指示灯开关。

设备接收修改设备日志级别与指示灯开关的指令，M2M服务器 ⇒ 设备，MQTT PUBLISH。主题为*app2dev/<DID>/<AppClientId>* (<DID>指设备的DID，<AppClientId>指发出消息的App的当前MQTT Client Identifier，<AppClientId>在设备回复时用到，见下文)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	可变长度	1~4	len(Flag...设置值bit0~bit7)
3	Flag	1	0x00
4	命令字	2	0x0010
5	设置值bit8~bit15	1	bit8 ~ bit15从低位(bit)向高位排列
6	设置值bit0~bit7	1	bit0 ~ bit7从低位(bit)向高位排列，与上面的bit8 ~ bit15一共组成bit0 ~ bit15，各位的定义如下： ● bit0: Error日志级别的开与关，0为关，1为开 ● bit1: Warning日志级别的开与关，0为关，1为开 ● bit2: Info日志级别的开与关，0为关，1为开 ● bit3: WiFi模组所有指示灯的总开关，0为关，1为开 ● bit4~bit15：保留

设备返回操作确认，设备 ⇒ M2M服务器，MQTT PUBLISH。主题为*dev2app/<DID>/<AppClientId>* (<DID>指设备的DID，<AppClientId>指接收消息的App的当前MQTT Client Identifier，<AppClientId>是请求指令在主题中带过来的，见上文)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	可变长度	1~4	len(Flag...命令字)
3	Flag	1	0x00
4	命令字	2	0x0011

4.11. 设备请求在线App的数量

设备在上电后可请求当前与设备有绑定关系的在线App的数量。

设备请求在线App的数量情况，设备 ⇒ M2M服务器，MQTT PUBLISH。主题为*ser2cli_req*。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	命令字	2	0x020F

4.12. 通知设备在线App数量的变化

当与设备有绑定关系的在线App数量发生变化或设备请求该数量时，云端向设备发送在线App的数量。

设备接收在线App的数量情况，M2M服务器 ⇒ 设备，MQTT PUBLISH。主题为 *ser2cli_res/<DID>* (<DID>指设备的DID)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	命令字	2	0x0210
3	在线App的数量	2	非负整数，大端字节序

4.13. 设备向云端及App发送日志

当设备相关的日志级别打开后，设备会主动向连接了的App及云端发送日志。

设备向App及云端发送日志，设备 ⇒ M2M服务器，MQTT PUBLISH。主题为*dev2app/<DID>* (<DID>指设备的DID)。

序号	字段名称	字节长度(B)	内容说明
1	固定包头	4	0x00000003
2	可变长度	1~4	len(Flag...日志内容)
3	Flag	1	0x00
4	命令字	2	0x0012
5	日志级别	1	1为Error，2为Warning，3为Info,
6	标签长度	2	len(标签)
7	标签	max 256	指示日志种类与关键字
8	来源长度	2	len(来源)
9	来源	max 32	指示日志是从哪里来的

10	日志内容长度	2	len(日志内容)
11	日志内容	max 65535	日志的内容，为UTF8编码的字符