海为 MQTT 数据互联
技术文档

Haiwell 海为®

厦门海为科技有限公司

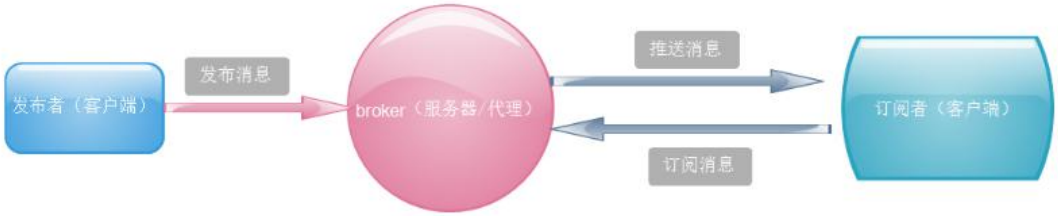
2019 年 07 月　　研发中心（研发三部）

1. 文档变更记录

序号	变更内容说明	版本号	版本日期	执笔人
1	初稿	0.1	2019/07/10	刘毅
2	修改终端标识(terminal identifier)为终端编号(terminal code)	0.2	2019/08/12	刘毅
3	互联工具订阅通配符 # 改为 +	0.3	2019/08/20	刘毅
4	补充关键字解释；示例补充数据格式和主题	0.4	2019/09/29	刘毅
5	新增互联工具云数据中心远程写入的使用规则	0.5	2019/10/09	刘毅
6	将 HMI 更改为物联终端,增加第三方软件数据对接的说明	0.6	2019/10/28	林义

2. MQTT 的介绍

MQTT 是一个轻量级协议,基于 TCP/IP 协议的发布(Publish)/订阅(Subscribe)消息转发模式，在物联网应用中大规模使用。 MQTT 协议的中心是 broker（ 服务器/代理） ，客户端通过订阅消息和发布消息进行数据交互 ，如下图所示：



那么如何才能订阅到我想要的数据呢？这就需要主题（Topic）的参与了。发布消息时会附带一个主题，只有订阅者订阅这个相同的主题，就会从 broker 收到消息。

同时，MQTT 带有主题过滤规则。

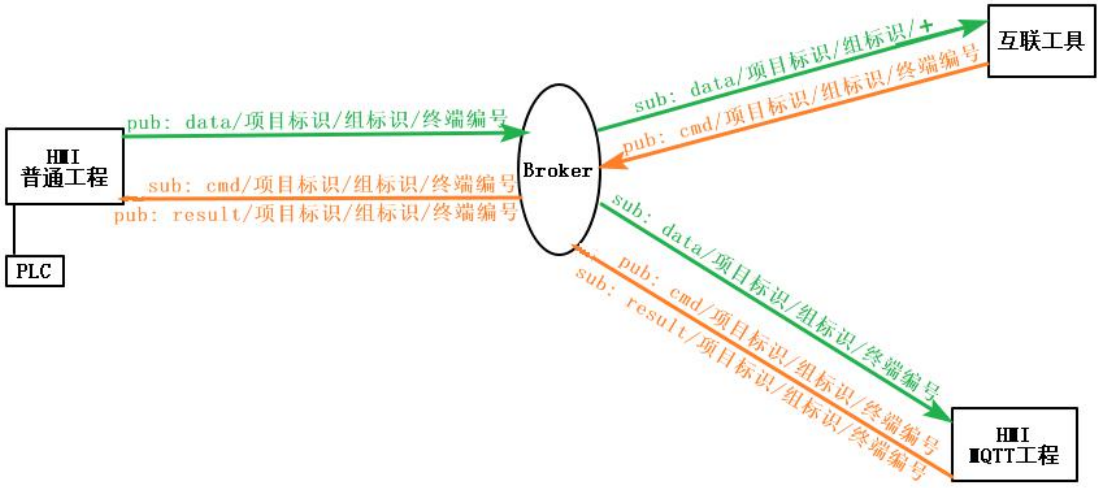
主题层级分隔符“/”，主题层级分隔符使得主题名结构化。如果存在分隔符，它将主题名分割为多个主题层级。斜杠（‘/’）用于分割主题的每个层级，为主题名提供一个分层结构。比如，

```
room212/electric
room212/tv/contrl/sensor
room212/tv/contrl/light
room212/air/sensor
```

多层通配符“#”，这是用于匹配主题中任意层级的通配符。多层通配符表示它的父级和任意数量的子层级。多层通配符必须位于它自己的层级或者跟在主题层级分隔符后面。不管哪种情况，它都必须是主题过滤器的最后一个字符。比如，如果客户端订阅主题 “china/xiangtan/#”，它会收到使用下列主题名发布的消息：

```
china/xiangtan
china/xiangtan/yuhu
china/xiangtan/yuetan/hnie
china/xiangtan/jiuhua/jiakao/kemusan
```

3. 数据互联简介



物联终端设备（HMI/CBOX/IPC）和互联工具之间的数据互联皆是通过 MQTT 协议来完成的，通过它们之间的配合，可以实现，

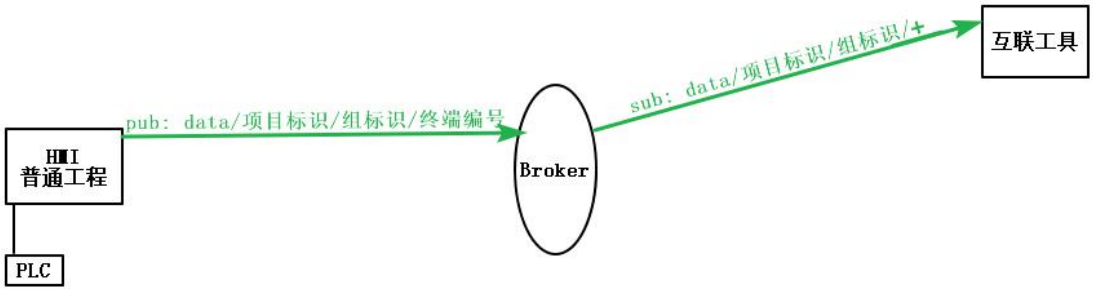
1. **数据上报及数据库存储（实现历史记录存入数据库）**
2. **多设备的可异地集中式控制（即远程写入）**
3. **第三方软件的数据对接（MQTT 对接或者数据库对接）**

上图主要展示了各设备和互联工具之间基于 MQTT 协议实现的主题订阅和发布的结构。其中的 sub 和 pub 分别是订阅（subscribe）和发布（publish）的缩写，紧跟其后的是**主题格式**。

- **项目标识**，用以区分不同的项目，也为了让多个项目可以共用一个 Broker。
- **组标识**，一个工程，可能有多个数据组，用以区分数据组。
- **终端编号** 这里的终端你可以简单理解成物联终端，一个组态工程可能需要下载到多台物联终端上，这个编号就是用来区分物联终端的，从而实现精准的定位。这是一个系统变量$TerminalCode，若值为空，主题中会自动补充为 PN 码。
- data，意寓数据的意思。
- cmd，意寓指令的意思。
- result，意寓 cmd 执行成功与否返回的结果。

下面就如何实现 **数据上报及数据库存储** 、 **远程写入**作详细的阐述，而第三方软件的数据对接，在介绍的过程中也有做了说明。

4. 数据上报及数据库存储



物联终端（HMI/CBOX/IPC/SCADA）中的工程的“数据组”功能，是一个即可以实现本地历史记录，也可以实现数据上报的功能，通过数据组的配置，可以实现定时上报、触发上报等数据上报机制。

只要数据组被触发上报，物联终端就会把数据发送到 MQTT 服务器，互联工具同时会从 MQTT 服务器取得这份数据，并写入数据库，**如果有第三方软件需要通过 MQTT 获取数据，则可以模拟互联工具，订阅相同主题，即可获取到数据**。我们直接以一个具体的例子来讲解下上述的过程。

假设现有一台物联终端，终端编号为“my_terminal”，其内的工程项目标识为“my_project”，数据组组名为“my_group_name”，组标识为“my_group_identifier”，包含两个变量，

通道标识	类型	当前值
Y0	开关型	1
V0	整型	99

注：**通道标识** 就是用以区分不同的变量，下同。

当被触发时，物联终端就会发布数据，主题为“data/my_project/my_group_identifier/my_terminal”，

```
{
    "_terminalTime" : "2018-01-01 06:30:30",
    "_groupName" : "my_group_name",
    "Y0" : "1",
    "V0" : "99"
}
```

内容格式是 **json**，其中**_terminalTime** 由物联终端在发布时自动生成，它表示发布的时间。**_groupName** 也是由物联终端自动生成，它表示数据组名。注意，为方便处理，json 中所有条目的数据格式都是字符串。你可以看到，虽然 V0 变量是整型，但是在 json 中的数据格式是字符串。

互联工具在进行对应配置后，会订阅主题“data/my_project/my_group_identifier/+”并取得这份数据**（如果第三方软件需要 MQTT 数据，可以订阅该主题）**，然后互联工具会创建一张名为“my_group_identifier”的数据表**（如果第三方软件需要数据，也可以读取该数据表）**，并把得到的数据写进数据库，数据库中的具体形式是这样的，

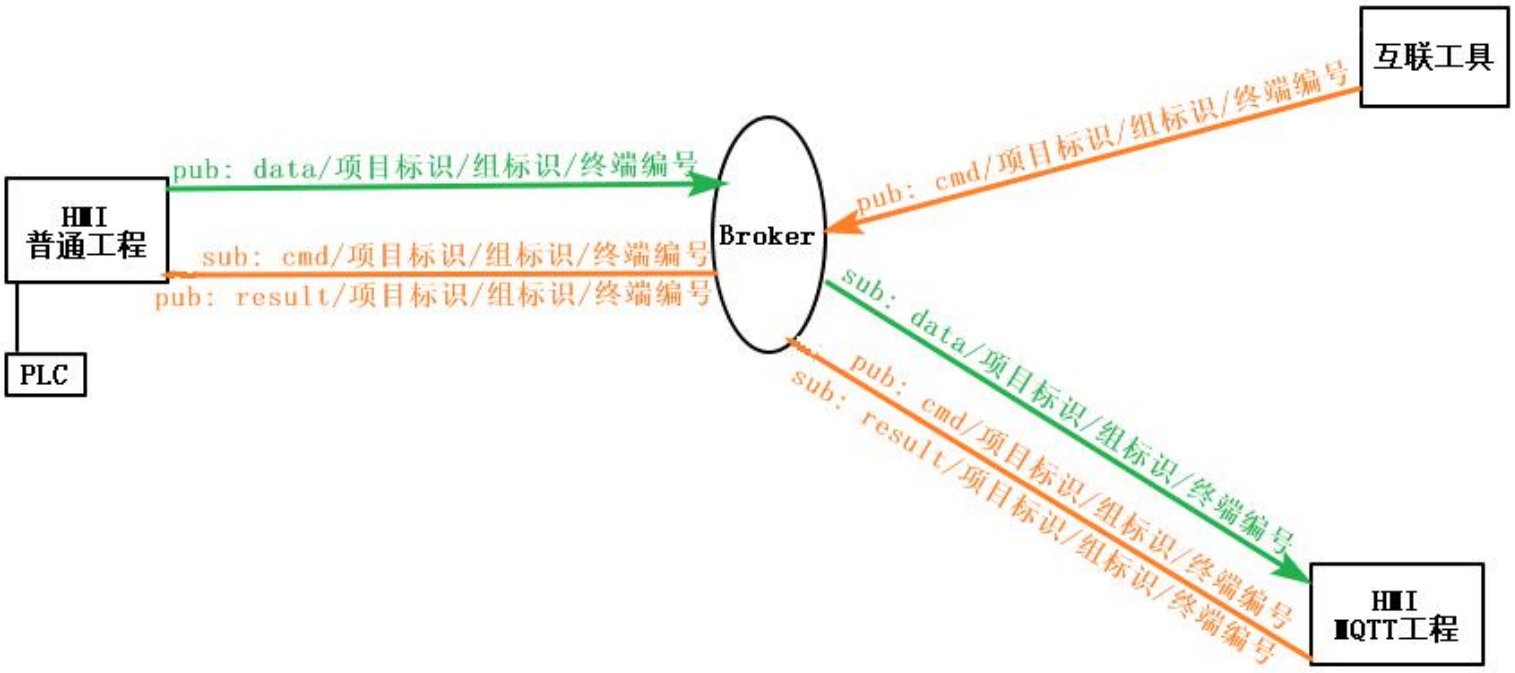
_id	_dbTime	_terminalTime	_groupName	Y0	V0
1	2018-01-01 06:30:35	2018-01-01 06:30:30	my_group_name	0	99

其中，新加入的_id 和_dbTime 由互联工具在写入数据库的时候自动生成。**_id** 是一个整型自增的主键，**_dbTime** 表示写进这条数据时当前数据库的时间。

各个字段的类型如下表，

字段名	数据类型(MySQL)	数据类型(Microsoft SQL Server)
_id	int unsigned primary key auto_increment	int identity (1,1) primary key
_dbTime	datetime	datetime2(3)
_terminalTime	varchar(50)	varchar(50)
_groupName	varchar(50)	varchar(50)
Y0	varchar(50)	varchar(50)
V0	varchar(50)	varchar(50)

5. 远程写入



绿色部分，普通工程发布的数据内容和格式已在上面阐述过，MQTT 工程订阅拿到数据后，会进行解析，并显示在对应变量的图元上。

橙色部分，即远程写入的部分。MQTT 工程和互联工具都可以进行远程写入的功能。**注意，要想该变量能被远程控制写入，必须在普通工程的数据组里将该变量勾选为“允许远程写入”。**

对于 MQTT 工程而言，当界面的变量被改动，会生成一条消息，并进行发布。举个例子，现有一个 MQTT 工程，项目标识为“my_project”，数据组组标识为“my_group_identifier”，设备终端编号为“my_terminal”。有一个变量的值被改变，该变量信息如下，

通道标识	类型	新值
V0	整型	99

如果要远程写入变量的值，则需要向 MQTT 服务器发布的主题为“cmd/my_project/my_group_identifier/my_terminal”，内容为：

```
{
    "type": "set_var",
    "payload": {
        "V0": "99"
    }
}
```

数据格式为 json。“type”表示 cmd 命令的类型，“set_var”表示修改变量，“payload”是要修改的变量通道标识和要修改的值。
如果是第三方软件需要通过 MQTT 实现数据写入，则发布该条指令即可。

普通工程收到后，修改本地变量，修改成功与否，会通过 result 前缀主题反馈给 MQTT 工程，MQTT 工程收到后，若判断到已成功修改，就会把该变量对应的新值更新到界面。以上为例，若普通工程成功修改本地的 V0 变量，则会发布主题为“result/my_project/my_group_identifier/my_terminal”的消息，

```
{
    "success": 1,
    "type": "set_var",
    "payload": {
        "V0": "99"
    }
}
```

success 为 0 表示失败，但目前，修改失败不会发布数据，只在成功时发布。

对于互联工具而言，用户需要在**任务设置**里启用**远程写入**，才能使用该功能。

一旦被启用，互联工具会自动在当前数据库下创建两张数据表：_command 和 _command_history（此表名为默认值，可由用户自定义）。
如果第三方软件需要通过数据库来实现远程数据写入，则根据规范对_command 写入数据即可。

_command 是一张命令表，互联工具启动后会以一定频率轮询这张表，并执行里边的命令，一旦命令执行成功，就会把这条命令移动到_command_history 表中，

并在_command 表中把它删掉。

现在我们来看下_command 和_command_history 数据表中的字段格式，

字段名称	类型	NULL	说明
id	整型	NOT NULL	自增主键
command_id	字符串	NULL	该命令的 id 标识，可由用户自行指定，可为空
project_identifier	字符串	NOT NULL	项目标识，不可为空
group_identifier	字符串	NOT NULL	组标识
terminal_code	字符串	NOT NULL	终端编号
type	字符串	NOT NULL	该命令的执行类型
channel_identifier	字符串	NULL	通道标识
value	字符串	NULL	通道标识的值
payload	字符串	NULL	channel_identifier 和 value 的第二种选择

两张表的字段名称和类型一致，如上表所示。

关于 payload 字段，需要特别说明下。考虑到一条命令可以发布多个变量，所以可以用 payload 字段来实现它，其格式为“Y0,1;V0,99”，通道标识和其对应的值以英文字符逗号隔开，多个变量之间以英文字符分号隔开。

如果选择云数据中心，那么 project_identifier 填入你的项目 id 即可。例如，项目 id 为“15826”，则 project_identifier 填“15826”。

下面我们以一个具体的例子来详细讲解下。假如现在_command 数据表中有两条命令，如下，

id	command_id	project_identifier	group_identifier	terminal_code	type	channel_identifier	value	payload
1		demo	group1	ter1	set_var	Y0	0	
2		demo	group2	ter2	set_var			Y0,1;V0,99

互联工具轮询这张表取得这两条命令后，会以服务质量为 2 进行发布，若命令发布成功（注意，发布成功的意思是可以保证 broker 收到这条命令，但不能保证你要写入的那个变量的值可以成功被更改），会把该命令移动到_commadn_history 数据表中，并从_command 表中删除该命令。

以上述两条命令为例，发布（Publish）的主题（Topic）和具体内容如下，

```
# 主题：cmd/demo/group1/ter1

# 上报的内容为（json 文本）：
{
```

```
    type: "set_var",
    payload: {
        "Y0": "0"
    }
}
```

```
# 主题：cmd/demo/group2/ter2

# 上报的内容为（json 文本）：
{
    type: "set_var",
    payload: {
        "Y0": "1",
        "V0": "99"
    }
}
```