污染物在线监控（监测）系统数据传输补充协议

Supplementary Protocol for Data Transmission of Pollutant Online Monitoring (Monitoring) System

上海建工集团工程研究总院 信息技术研究所

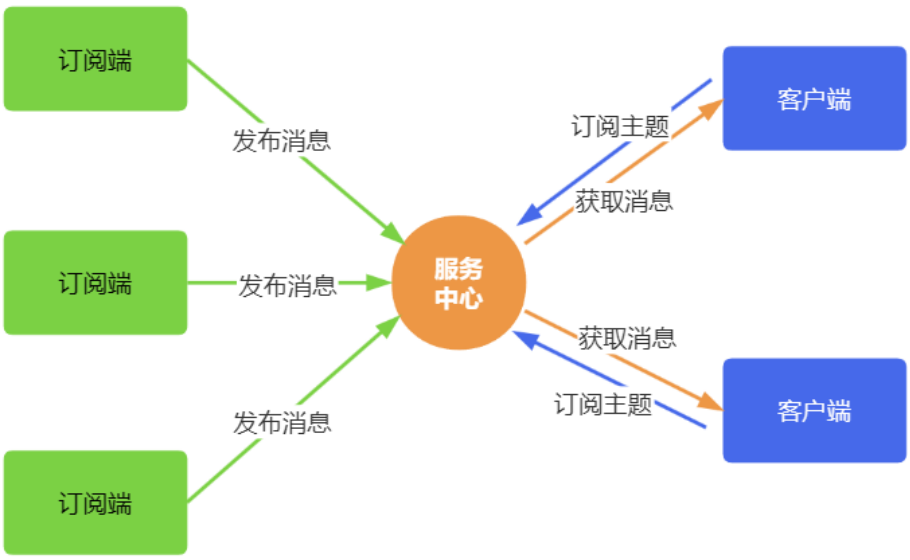
2024-04-30

目录

概述

MQTT(消息队列遥测传输)是 ISO 标准(ISO/IEC PRF 20922)下基于发布/订阅范式的消息协议。

MQTT 是机器对机器(M2M)/物联网(IoT)连接协议。它被设计为一个极其轻量级的发布/订阅消息传输协议。对于需要较小代码占用空间和/或网络带宽非常宝贵的远程连接非常有用，是专为受限设备和低带宽、高延迟或不可靠的网络而设计。这些原则也使该协议成为新兴的“机器到机器”(M2M)或物联网(IoT)世界的连接设备，以及带宽和电池功率非常高的移动应用的理想选择。例如，它已被用于通过卫星链路与代理通信的传感器、与医疗服务提供者的拨号连接，以及一系列家庭自动化和小型设备场景。它也是移动应用的理想选择，因为它体积小，功耗低，数据包最小，并且可以有效地将信息分配给一个或多个接收器。



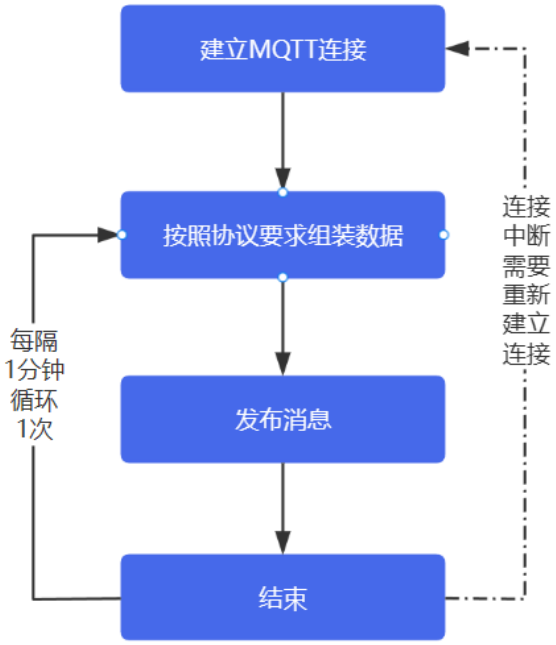
（图 1）

1、使用说明

1.1 数据协议：

《污染源在线自动监控（监测）系统数据传输标准》（HJT 212-2017）

数据格式和内容，需要严格遵循如上协议。



（图 2）

消息体内容示例：

```
##0589QN=20240422104800000;ST=22;CN=2051;PW=123456;MN=HJKJ-24-XH-00002;CP=&&DataTime=20240422104800;a34001-Avg=0.051,a34001-Max=0.051,a34001-Min=0.051,a34001-Flag=N;a01007-Avg=0.0,a01007-Max=0.0,a01007-Min=0.0,a01007-Flag=N;a01008-Avg=180.0,a01008-Max=180.0,a01008-Min=180.0,a01008-Flag=N;a01001-Avg=19.9,a01001-Max=19.9,a01001-Min=19.9,a01001-Flag=N;a01002-Avg=78.7,a01002-Max=78.7,a01002-Min=78.7,a01002-Flag=N;a01006-Avg=102.6,a01006-Max=102.6,a01006-Min=102.6,a01006-Flag=N;a50001-Avg=51.0,a50001-Max=51.0,a50001-Min=51.0,a50001-Flag=N;cpm-Avg=0.041,cpm-Max=0.041,cpm-Min=0.041,cpm-Flag=N;&&6E80
```

2、调用地址及身份

正式环境会按照项目、或设备供应商的不同，下发不同的身份信息给到设备供应商进行开发标识。测试环境使用统一身份进行测试联调。

1）测试环境

项	内容
订阅地址	ssl://47.117.118.169:18883
订阅主题	monitor/pole/
用户名	monitor_pole
密码	Monitor.2024#Pole
ClientID	设备序列号(MN)

（表 1）

2）正式环境

项	内容
订阅地址	待定
订阅主题	待定
用户名	待定
密码	待定
ClientID	设备序列号(MN)

（表 2）

3、使用示例:

2.1 Java 代码示例:

1)、需要添加 Maven 依赖项到项目中:

```
<dependency>
    <groupId>org.eclipse.paho</groupId>
    <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
    <version>1.2.5</version>
</dependency>
```

2)、连接到 MQTT 代理并发布消息:

```
import org.eclipse.paho.client.mqttv3.*;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;
public class MqttExample {
    public static void main(String[] args) {
        try {
            // 创建 MQTT 客户端实例，使用内存存储持久化客户端配置
            MqttClient client = new MqttClient("tcp://broker.hivemq.com:1883",
MqttClient.generateClientId(), new MemoryPersistence());
            // 创建 MQTT 连接参数，包括用户名和密码（如果需要）
            MqttConnectOptions options = new MqttConnectOptions();
            options.setUserName("your_username");
            options.setPassword("your_password".toCharArray());
            // 连接到 MQTT 代理
            client.connect(options);
            // 发布消息到主题“test/topic”
            String payload = "Hello, MQTT!";
            MqttMessage message = new MqttMessage(payload.getBytes());
            message.setQos(2); // 设置服务质量为 2（确保传递）
            client.publish("test/topic", message);
        } catch (MqttException e) {
            e.printStackTrace();
        }
    }
}
```

2.2 C#代码示例:

1）连接 MQTT 服务器

```
using MQTTnet;
using MQTTnet.Client;
using MQTTnet.Client.Options;

var factory = new MqttFactory();
var client = factory.CreateMqttClient();
var options = new MqttClientOptionsBuilder()
    .WithTcpServer("localhost", 1883)
    .WithClientId("client1")
    .Build();
await client.ConnectAsync(options);
```

2）发布 MQTT 消息

```
var message = new MqttApplicationMessageBuilder()
    .WithTopic("topic1")
    .WithPayload("Hello MQTT")
    .WithQualityOfServiceLevel(MqttQualityOfServiceLevel.AtMostOnce)
    .WithRetainFlag(false)
    .Build();

await client.PublishAsync(message);
```