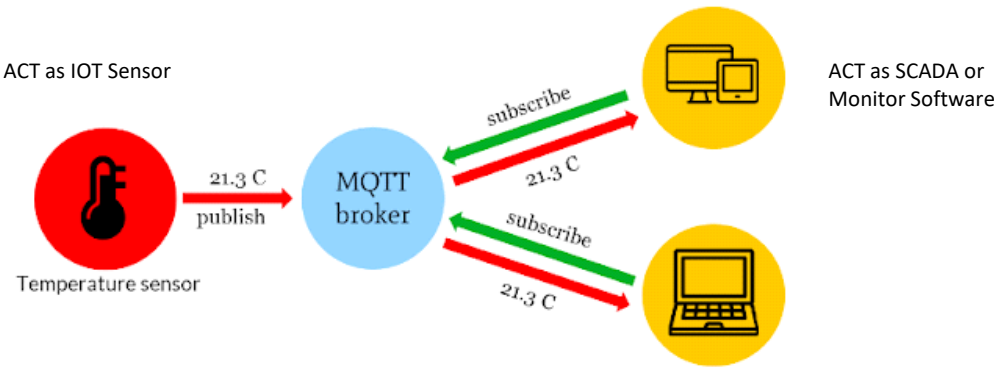
MQTT Definition

Friday, June 26, 2020 7:40 PM

MQTT (MQ Telemetry Transport or Message Queuing Telemetry Transport) is an open publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP



Schematic data flow from sensor (machine) to devise (machine)

Roles of Client (Publisher)

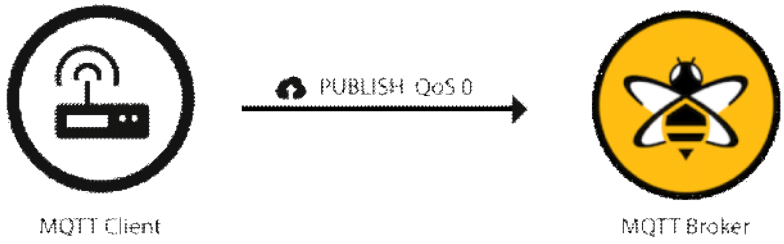
- Initiate Communication with the Broker with unique Id
- Send (Publish) information based on specified period of time
- Log any action

Roles of Client (Subscriber)

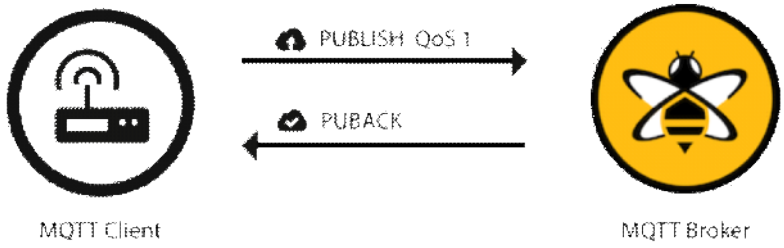
- Initiate Communication with the Broker with unique Id
- Wait and listen for any incoming information (Subscribe)
- Log any action

General Roles of Communication between clients

- At most once (fire and forget) (Level 0)
  - the message is sent only once and the client and broker take no additional steps to acknowledge delivery .

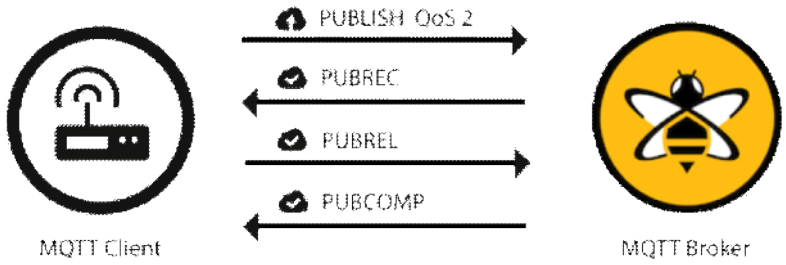


- At least once (acknowledged delivery) (Level 1)
  - the message is re-tried by the sender multi



ple times until acknowledgement is received .

- Exactly once (assured delivery) (Level 2)
  - the sender and receiver engage in a two-level handshake to ensure only one copy of the message is received .



Detailed Description

Online Brokers

Friday, June 26, 2020 7:39 PM

Online Brokers Types:

▹ Private MQTT Broker

Name	Broker Address	TCP Port	TLS Port	WebSocket Port	Message Retention
Eclipse	mqtt.eclipse.org	1883	N/A	80, 443	YES
Mosquitto	test.mosquitto.org	1883	8883, 8884	80	YES
HiveMQ	broker.hivemq.com	1883	N/A	8000	YES
Flespi	mqtt.flespi.io	1883	8883	80, 443	YES
Dioty	mqtt.dioty.co	1883	8883	8080, 8880	YES
Fluux	mqtt.fluux.io	1883	8883	N/A	N/A

▹ Public MQTT Broker

Name	Link	TCP Port	TLS Port	WebSocket Port	Message Retention	Persistent Session
Azure	Link	NO	8883	443	NO	Limited
AWS	Link	NO	8883	443	NO	Limited
CloudMQTT	Link	Custom Port	Custom Port	Custom Port	NOT SURE	YES

Detailed Description

Example on Private MQTT Broker

Server	Username	Password	Prot	SSL Port	Websockets Port (TLS only)
tailor.cloudmqtt.com	jxdgnsju	by2uhfp1iSx8	11300	21300	31300

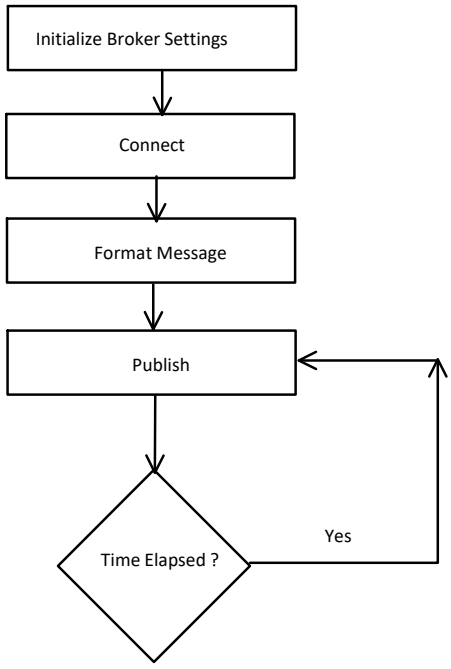
Steps

Friday, June 26, 2020 10:50 PM

1. Open Visual Studio
2. Create Console Application Project "Publisher" under solution "MQTTPubSub"
3. Add Console Application Project "Subscriber" under the same solution
4. Add Shared Application Project "MQTTShared" under the same solution
5. Add "MQTTShared" Reference to both the project of "Publisher" and "Subscriber"
6. Start Coding

Publisher [Script]

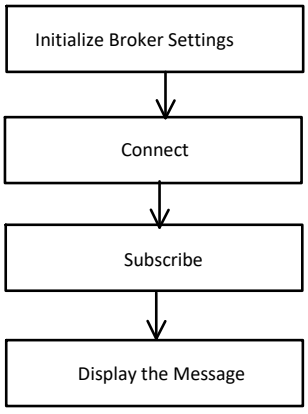
Friday, June 26, 2020 10:28 PM




```
using MQTTShared;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using uPLibrary.Networking.M2Mqtt;
namespace Publisher
{
    class Program
    {
        static void Main(string[] args)
        {
            //Intailize the Broker Settings
            BrokerModel _Broker = new BrokerModel();
            _Broker.Url = "tailor.cloudmqtt.com";
            _Broker.Username = "jxdgnsju";
            _Broker.Password = "by2uhfp1iSx8";
            _Broker.Port = 11300;
            _Broker.SSLPort = 21300;
            _Broker.WebSocketPort = 31300;
            //Connect to Server
            MQTT mQTT = new MQTT();
            mQTT.Initailize(_Broker);
            mQTT.Connect();
            while (true)
            {
                if (mQTT.Connected)
                {
                    //publish string
                    //mQTT.Publish("StringValue", "5");
                    //publish json Object
                    Sensor sensor = new Sensor();
                    var result = sensor.CreateSensor();
                    string Jsonresult = JsonConvert.SerializeObject(result);
                    mQTT.Publish("SensorObject", Jsonresult);
                    //publish json List of Objectss
                    //var resultList = sensor.CreateSensorList();
                    //string JsonresultList = JsonConvert.SerializeObject(resultList);
                    //mQTT.Publish("SensorList", JsonresultList);
                }

                Thread.Sleep(10000);
            }



        }
    }
}
```

Subscriber [Script]

Friday, June 26, 2020 10:30 PM




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MQTTShared;
using Newtonsoft.Json;
using uPLibrary.Networking.M2Mqtt.Messages;
namespace Subscriber
{
    class Program
    {
        static void Main(string[] args)
        {
            //Intailize the Broker Settings
            BrokerModel _Broker = new BrokerModel();
            _Broker.Url = "tailor.cloudmqtt.com";
            _Broker.Username = "jxdgnsju";
            _Broker.Password = "by2uhfp1iSx8";
            _Broker.Port = 11300;
            _Broker.SSLPort = 21300;
            _Broker.WebSocketPort = 31300;
            //Connect to Server
            MQTT mQTT = new MQTT();
            mQTT.Initailize(_Broker);
            mQTT.Connect();
            mQTT.MqttMsgPublishReceived +=client_MqttMsgPublishReceived;
            mQTT.Subscribe( "StringValue");
            mQTT.Subscribe("SensorObject");
            //mQTT.Subscribe("SensorList");
        }
        static void client_MqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
        {

                //Return string
                var RecievedMsg = Encoding.UTF8.GetString(e.Message);
            //Return json Object
           // var RecievedObj = JsonConvert.DeserializeObject<Sensor.Data>(RecievedMsg);
            //Return json List of Object
            //var RecievedObj = JsonConvert.DeserializeObject<List<Sensor.Data>>(RecievedMsg);

            Console.WriteLine("Received = " + RecievedMsg + " on topic: " + e.Topic + " at " + DateTime.Now);

        }
    }
}
```

MQTT [Script]

Saturday, June 27, 2020 2:35 AM

```
using System;
using System.Collections.Generic;
using System.Text;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using static uPLibrary.Networking.M2Mqtt.MqttClient;
namespace MQTTShared
{
   public class MQTT
    {
        private BrokerModel _Broker = new BrokerModel();
        private MqttClient _client { get; set; }
        public event MqttMsgPublishEventHandler MqttMsgPublishReceived;
        public bool Connected;
        public bool Initailized;
        public void Initailize(BrokerModel Broker)
        {

            if (Broker != null)
            {
                if (!string.IsNullOrEmpty( Broker.Url)
                    && !string.IsNullOrEmpty(Broker.Username)
                    && !string.IsNullOrEmpty(Broker.Password)
                    && Broker.Port > 0
                    && Broker.SSLPort > 0
                    && Broker.WebSocketPort > 0)
                {
                    _Broker = Broker;
                    Initailized = true;
                    Console.WriteLine("MQTT is succesfully intialized");
                }
                else
                {
                    Console.WriteLine("MQTT is badly intialized");
                }

            }
            else
            {
                Console.WriteLine("MQTT is badly intialized");
            }
        }
        public void Connect()
        {
            if (Initailized)
            {
                _client = new MqttClient(_Broker.Url, _Broker.Port, false, MqttSslProtocols.None, null, null); ;
                string clientId = Guid.NewGuid().ToString();
                Console.WriteLine("Client " + clientId + " started");
                try
                {
                    _client.Connect(clientId, _Broker.Username, _Broker.Password);
                    Connected = true;
                    Console.WriteLine("MQTT is succesfully Connected");
                }
                catch (Exception e)
                {
                    Console.WriteLine("Client " + clientId + " cannot connect");
                    Console.WriteLine(e.Message);
                }
            }



        }
```

```
        public void Publish(string Topic, string Message)
        {
            if (Connected)
            {
                string strValue = Convert.ToString(Message);
                // publish a message on a topic with QoS 2
                _client.Publish(Topic, Encoding.UTF8.GetBytes(strValue), MqttMsgBase.QOS_LEVEL_AT_MOST_ONCE,false);
                Console.WriteLine("Publish = " + Message + " is Published on Topic " +Topic+ " at " + DateTime.Now);
            }
            else
            {
                Console.WriteLine("Client " + _client.ClientId + " is not connected");
            }
        }
        public void Subscribe (string Topic)
        {
            if (Connected)
            {
                _client.MqttMsgPublishReceived += MqttMsgPublishReceived;
                _client.Subscribe(new string[] { Topic }, new byte[] { MqttMsgBase.QOS_LEVEL_AT_MOST_ONCE });
            }
            else
            {
                Console.WriteLine("Client " + _client.ClientId + " is not connected");
            }

        }
    }
}
```

BrokerModel [Script]

Saturday, June 27, 2020 2:36 AM

```
using System;
using System.Collections.Generic;
using System.Text;
namespace MQTTShared
{
    public class BrokerModel
    {
        public string Url { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public int Port { get; set; }
        public int SSLPort { get; set; }
        public int WebSocketPort { get; set; }
    }
}
```

SensorModel [Script]

Saturday, June 27, 2020 2:37 AM

```
using System;
using System.Collections.Generic;
using System.Text;
namespace MQTTShared
{
    public interface ISensor
    {
        Sensor.Data CreateSensor();
        List<Sensor.Data> CreateSensorList();
    }
    public class Sensor : ISensor
    {
        public class Data
        {
            public DateTime TimeStamp { get; set; }
            public int Value { get; set; }
            public int SensorId { get; set; }
            public string Unit { get; set; }
        }

        public Data CreateSensor()
        {
            return new Data
            {
                SensorId = new Random().Next(1000, 9999),
                TimeStamp = DateTime.Now,
                Value = new Random().Next(-100, 100),
                Unit = "Celsius"
            };
        }
        public List<Data> CreateSensorList()
        {
            List<Data> data = new List<Data>();
            data.Add(CreateSensor());
            data.Add(CreateSensor());
            data.Add(CreateSensor());
            data.Add(CreateSensor());
            data.Add(CreateSensor());
            return data;
        }
    }
}
```

Installed Packages

Friday, June 26, 2020 10:48 PM

M2Mqtt
Documentation

https://m2mqtt.wordpress.com/m2mqtt_doc/

Newtonsoft.Json