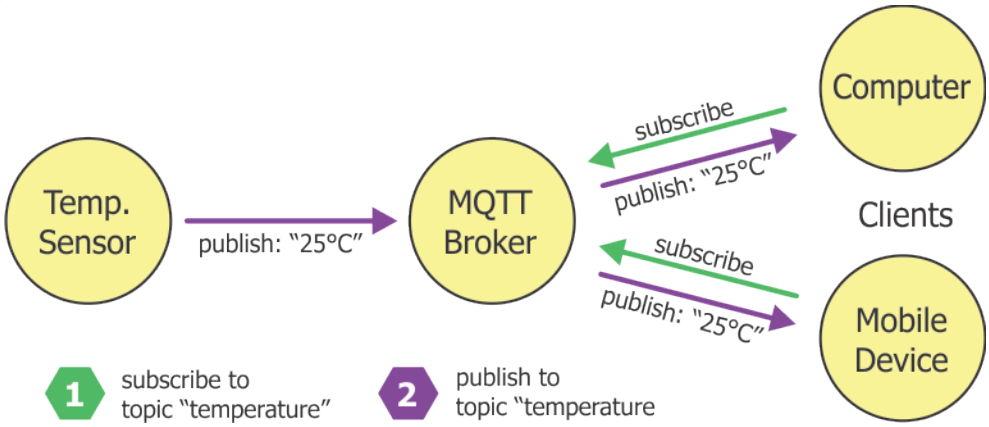
NEW! CIMScan IoT Hub with MQTT Access

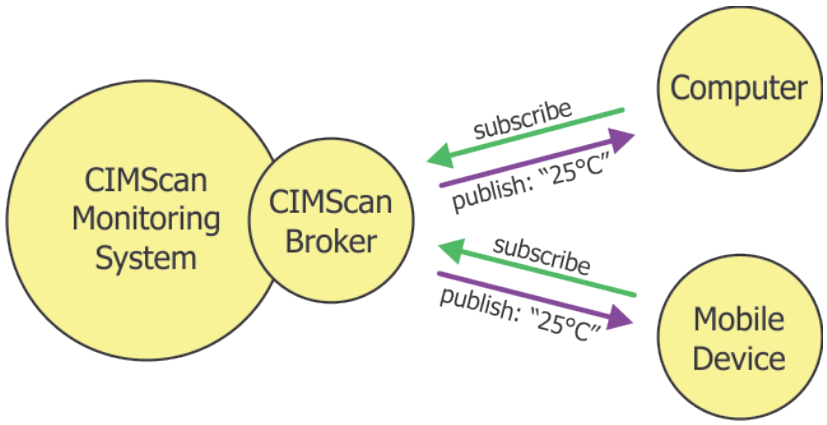
CIMScan cloud-based servers now support easy and secure access to their databases via MQTT. MQTT, short for Message Queue Telemetry Transport, is a lightweight protocol that uses a publish/subscribe architecture in contrast to HTTP with its request/response paradigm. Publish/Subscribe is event-driven and enables messages to be pushed to clients. The central communication point is the MQTT broker which is in charge of dispatching all messages between the senders and the rightful receivers. Each client that publishes a message to the broker, includes a topic into the message. The topic is the routing information for the broker. Each client which wants to receive messages subscribes to a certain topic, and the broker delivers all messages with the matching topic to the client. Therefore, the clients don't have to know each other; they only communicate over the topic. This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.

The Publish/Subscribe Architecture



The difference between MQTT and HTTP is that a client doesn't have to pull the information it needs, but the broker pushes the information to the client in case there is something new. Therefore, each MQTT client has a permanently open TCP connection to the broker. If this connection is interrupted by any circumstances, the MQTT broker can buffer all messages and send them to the client when it is back online.

Since the CIMScan Monitoring System contains the sensor information, the MQTT model looks like the diagram shown here:



As mentioned before, the central concept in MQTT to dispatch messages are topics. **A topic is a simple string that can have more hierarchy levels, which are separated by a slash.** A sample topic for sending temperature data of the living room could be **house/living-room/temperature**. On one hand, the client can subscribe to the exact topic, or on the other hand, use a wildcard. The subscription to **house/+/temperature** would result in all messages sent to the previously mention topic **house/living-room/temperature,** as well as any topic with an arbitrary value in the place of living room, for example, **house/kitchen/temperature**. The plus sign is a **single level wild card** and only allows arbitrary values for one hierarchy. If you need to subscribe to more than one level, for example to the entire subtree, there is also a **multilevel wildcard (#)**. It allows you to subscribe to all underlying hierarchy levels. For example **house/#** is subscribing to all topics beginning with **house**.

Available Topics

The CIMScan Monitoring system is, of course, the source for all the sensor information, and the CIMScan Broker is notified every time any of the measurement values are updated. All the Topics related to sensors begin with the following string …

Cimscan/DeptName/GroupName/Location/SensorName/

… with the tag strings shown in the table below appended to fully define the Topic.

Measurements	Statistics	Limits
Value	*MaxValue*	*LowAlarm*
ReadTime	*MinValue*	*HighAlarm*
AlarmStatus	*Time_Max*	*LowWarning*
SensorSerial	*Time_Min*	*HighWarning*
Units	*AvgValue*	
TagName	*CalcValue*	
MacAddress	*Time*	
AlarmString		
AlarmTimeStamp		

Remember that the wildcards (# and +) can be used anywhere in the base string to access multiple tags with a single topic.

Events generated at the Department level are published under the following Topic:

Cimscan/DeptName/Events = errors and other system-level events related to the Department
Cimscan/DeptName/Alerts = alert strings generated for monitoring points within the Department
Cimscan/DeptName/Alarms = entries in Alarm List for monitoring points within the Department

Each message can be published with one of three **Quality of Service Levels (QoS)**. These levels are defined when a subscription is issued by the client and is associated with different guarantees. A message send with level 0 doesn't have a guarantee at all, it implies fire and forget. Level 1 guarantees that the message will at least arrive once, but can arrive more than once. Level 2 is the most sophisticated

choice, which guarantees that the message arrives at the destination exactly once. The choice of QoS is a trade-off, between protocol overhead and the guarantee that the message arrives because ensuring QoS 2 is using more bandwidth than QoS 0.

The client can also define a **Last Will and Testament (LWT)** string and forward it to the Broker on a Topic by Topic basis. This string is sent to the client along with the Topic whenever the Broker detects that a failure has occurred with the measurement data that is being sent to the client.

Using the Library

A simple MQTT client library is provided to allow you to quickly and easily create an application that can request and receive MQTT data from the CIMScan Broker. First you have to create an instance of MqttClient class which provides only one mandatory parameter (the IP address or the host name of the broker you want to connect to) and some optional parameters with default values (MQTT broker port, secure connection and X.509 certificate). In the simpler case, you can use the default port (1883) and you don't have the support for secure connection based on SSL/TLS using default values for optional parameters and specifying only broker address (or host name).

```
MqttClient client = new MqttClient(IPAddress.Parse("192.168.10.53"));
```

Next, you need to create one or more event handlers and register them with the library.

```
MqttMsgPublishReceived = processed when QoS level 0 messages are received
MqttMsgSubscribed = processed when Broker completes subscription to a Topic
MqttMsgUnsubscribed = processed when Broker unsubscribes a Topic
MqttMsgPublished = processed QoS level 1 or 2 messages are received from the Broker
```

After you have registered all events you are interested in, you can use the Connect() method of MqttClient class to connect to the broker. The only mandatory parameter is the client ID that must be unique; the other parameters have some default values and they are related to:

- Username and password for client authentication (default values are null, no authentication);
- Will message feature (default values provides NO Will message);
- Clean session for removing subscriptions on disconnection (default value is true);
- Keep Alive period for keeping alive connection with ping message (default value is 60 seconds);

The connect method has the following structure.

```
client.Connect(Guid.NewGuid().ToString());
```

To subscribe and unsubscribe to a topic, the MqttClient class provides Subscribe() and Unsubscribe() methods. The former needs the list of topics and relative QoS levels to subscribe and the latter needs only the list of topic to unsubscribe. The Subscribe() method returns the message id for subscription.

```
client.Subscribe(topics, qosLevels);
client.Unsubscribe(topics);
```

For More Information Contact

CIMTechniques, Inc.

1215 Prince Street
Beaufort, SC 29902
(843) 532-9897
sales@cimtechniques.com