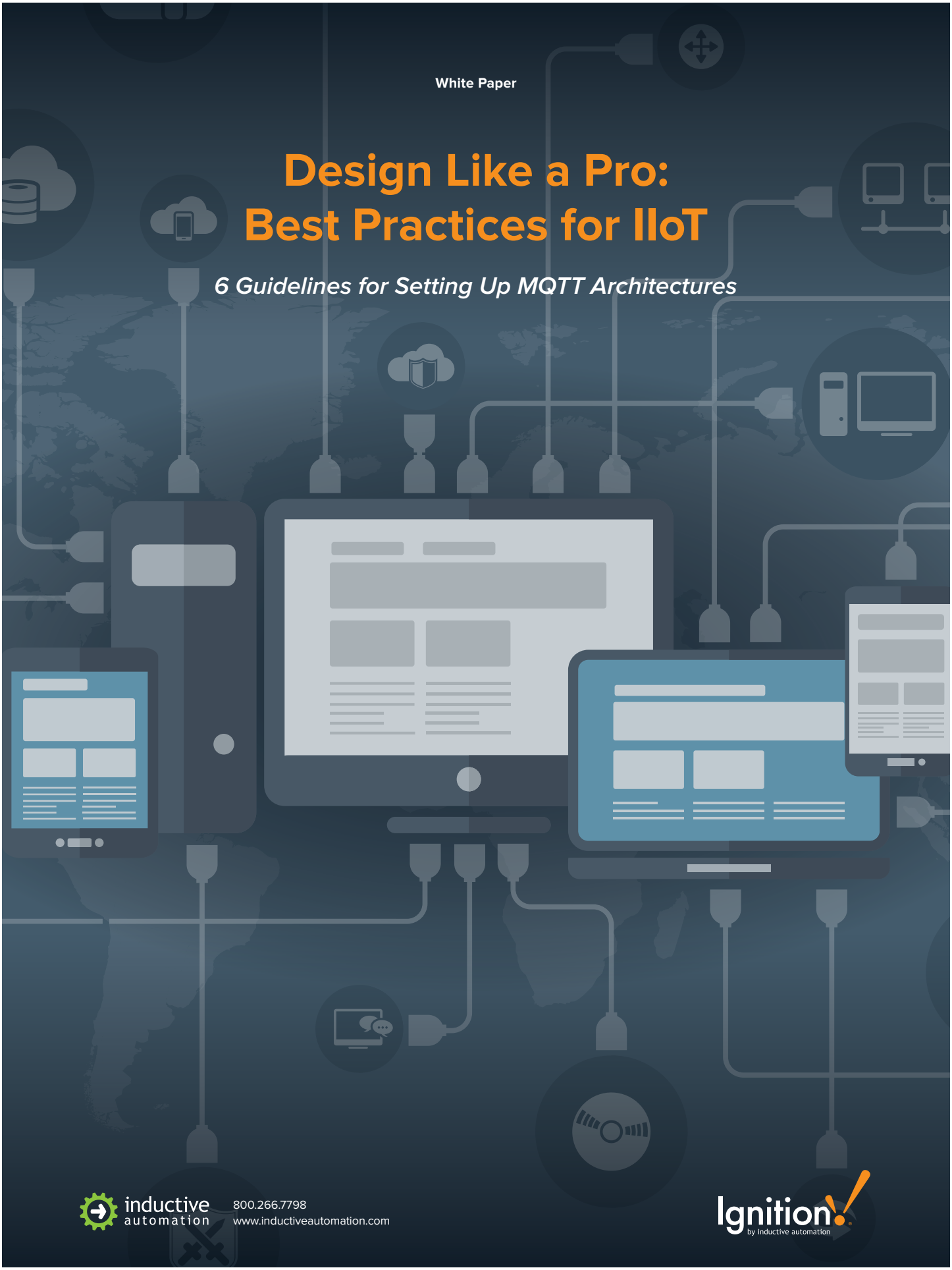
White Paper

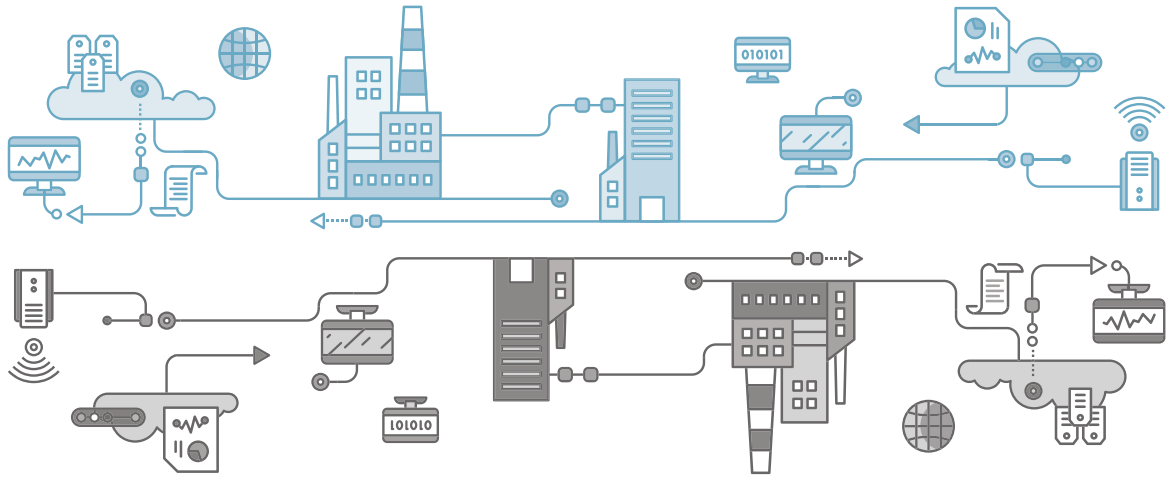
Design Like a Pro: Best Practices for IIoT

6 Guidelines for Setting Up MQTT Architectures



inductive automation
800.266.7798
www.inductiveautomation.com

Ignition! by inductive automation



The industrial automation industry is benefiting from the incredible opportunities made possible by the Internet of Things (IoT).

While the IoT has shown incredible promise within the corporate and consumer environment, the true value of the IoT can be found in the industrial space, which has aptly become known as the Industrial Internet of Things (IIoT).

The unquenchable thirst for industrial data has ignited this IIoT movement to bring operational technology (OT) up to speed with its enterprise counterpart, information technology (IT). This movement aims to overcome hurdles that have handicapped the growth of the industrial world, and to connect data across a wide network throughout an organization.

The IIoT unlocks data, clearing the way for easy access and shareability. By working with IT, OT can leverage the scalability and flexibility of open technologies to access and share all types of data with every level of an organization. In this white paper, we'll provide some best practices when approaching your new IIoT infrastructure.

Best Practice #1: Gradually Transition to a New IIoT Infrastructure

There is a lot of hype about the IIoT and many organizations want to leverage the benefits it promises. However, for many organizations, the path to technological adoption seems unclear, and some still question if IIoT will ever happen. Fortunately, with current and emerging offerings, organizations can actually take full advantage of IIoT today. Before making the leap, though, they should recognize that legacy devices are still in use. Planning and patience are required as you move forward with an IIoT solution for your organization. As the old saying goes, you should look before you leap.

Build a Parallel Infrastructure

There are still hundreds of millions of proprietary legacy PLCs and devices being used by organizations today, and they will continue to be in use for many years to come. Upgrading all of these devices would be incredibly cost-prohibitive. It would also be very difficult to just switch to a new technology, because making a quick switch could result in a catastrophic failure and loss of revenue.

Your best approach is to build a parallel infrastructure alongside with your existing installation and gradually transition devices from your old system to your new IIoT infrastructure. Many systems are

critical in nature and upgrades could cause outages, which are unacceptable. Building a system in parallel allows you and your organization to compare data from your established system with your new system. The gradual approach helps you to make sure your new system works and is stable before making a complete infrastructure transition.

Allow Time for Testing

Migrating to a brand-new IIoT system of a large magnitude requires thought and thorough planning. This is a process you shouldn't rush nor take lightly.

Taking your time affords you the ability to test your system. As you gradually build your new infrastructure, you can test along the way to make sure communication is stable. Furthermore, if a failure occurs while you install your new infrastructure, the current system is still available, mitigating any downtime.

As you begin developing a plan for your new IIoT infrastructure, you should start looking at which communication protocols to use in your infrastructure. The protocol you choose will determine the IIoT devices and software for your infrastructure. Take the time to understand the needs of your organization and how your current system is set up. The final solution you choose is a large commitment. Once implemented, your infrastructure will be in place for many years to come.

In the next section, we'll look at MQTT and why it is the ideal communications protocol for IIoT.

Best Practice #2: Choose MQTT as Your IIoT Messaging Protocol

Your entire IIoT solution will heavily depend on the protocol you choose, as it is the backbone of your system.

Most common IIoT protocols fall into two categories. One category is the publish-and-subscribe (pub-sub) protocols which connect and publish data to a topic on an intermediary broker. MQTT, AMQP, DDS, and XMPP are examples of pub-sub protocols.

The other category is the poll-response or client-server protocols, such as Allen-Bradley, Omron, and Modbus, in which clients continually connect to the server and make requests to determine if any data has changed.

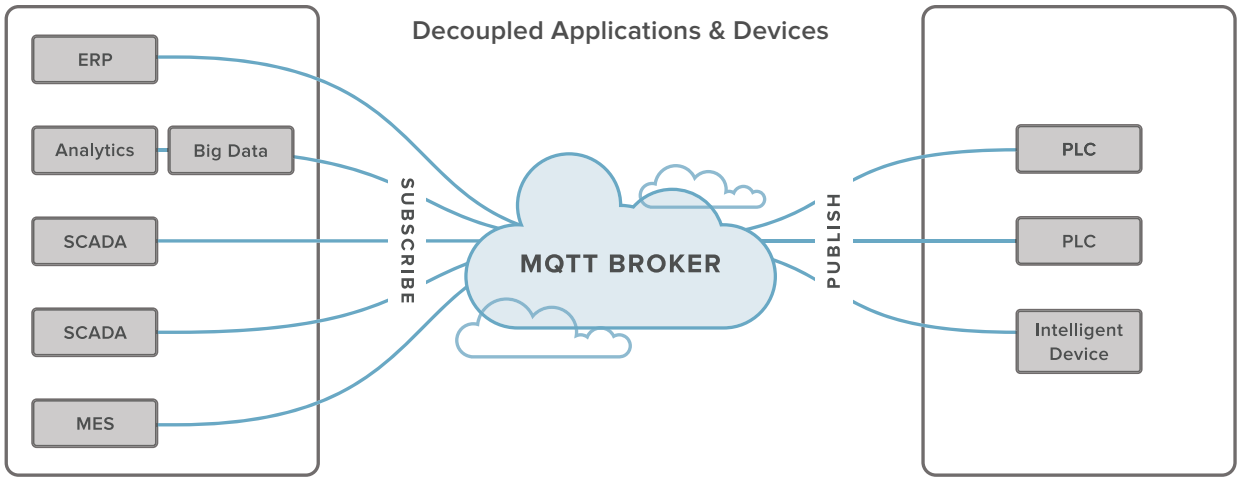
Of the two categories, which one should you choose? To effectively build a highly scalable solution with a high level of efficiency, it is best to adopt a publish-subscribe communication protocol. Rather than connecting applications directly to devices, publish-subscribe protocols decouple devices and allow applications to connect to middleware. Through middleware, you can connect any application that requires data from any device without placing any heavy demands on the network.

From the list of available protocols in the pub-sub category, we highly recommend using MQTT. More than just a protocol, MQTT is the foundation for building your new architecture, making IIoT a reality today.

Make MQTT Your Communications Protocol

While MQTT's recent emergence into the limelight may suggest that it's a brand new technology, MQTT has been around for quite some time. In 1999, Dr. Andy Stanford-Clark of IBM and Arlen Nipper invented a messaging protocol that was mainly intended for real-time, oil-and-gas SCADA systems. At the time, operational technology and information technology were two separate worlds. Unlike IT, bandwidth in OT was neither free nor unlimited.

In an effort to circumvent the communication limitations of OT, MQTT was designed to be a lightweight, pub-sub protocol that economizes on bandwidth. However, the true value of MQTT is now found in its ability to decouple edge devices from applications that need the data. Traditional poll-response communication protocols can eat a lot of bandwidth without providing any real value. MQTT's pub-sub method allows devices to put data on message-oriented middleware (MOM). Instead of applications constantly checking devices for any value changes, applications can connect to a MOM and subscribe to the important data that they need, including device state information.



Since MQTT has proven to be a formidable communications protocol, its use has gone far beyond the oil and gas industry, and it has emerged as the defacto standard for IIoT and M2M messaging. In the Eclipse Foundation's 2016 IoT Developer Survey, 80 percent of the respondents chose MQTT as the leading protocol for IIoT. MQTT is becoming more available as manufacturers begin to embed MQTT onto their devices. With so much interest in MQTT, it is safe to say that MQTT is the best choice for your IIoT solution.

What Makes MQTT the Ideal Protocol?

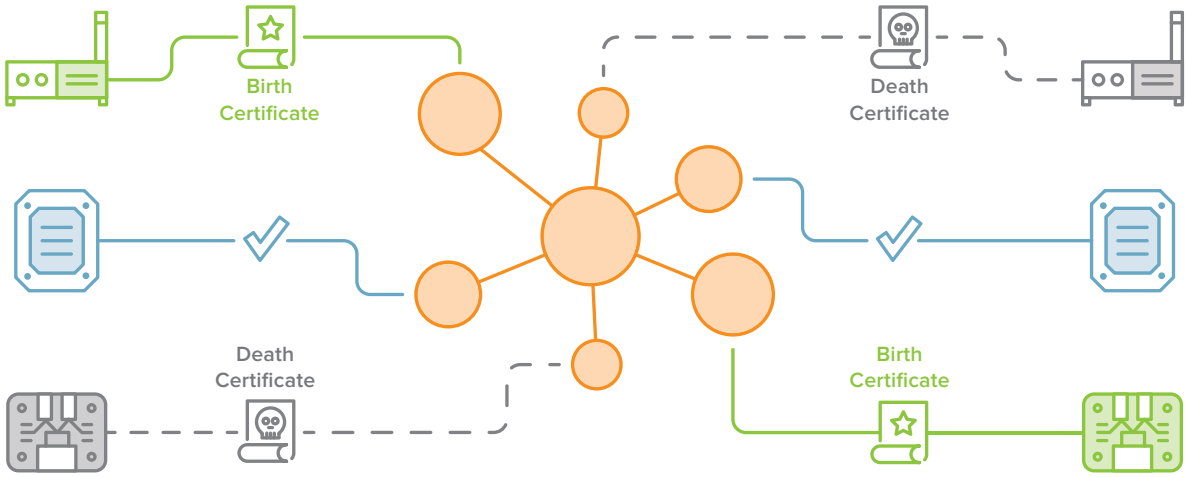
MQTT has three distinct features that also make it the ideal IIoT protocol: low bandwidth, TLS security, and stateful awareness.

Limited bandwidth presents a serious challenge to IIoT, especially for remote locations, which is why MQTT is the perfect solution. It is a lightweight, low-bandwidth communications protocol that uses a pub-sub methodology. Poll-response protocols send and receive a lot of repetitive data which can take up an unnecessary amount of bandwidth. MQTT employs a MOM which decouples devices from applications and thus reduces bandwidth usage. Devices connect directly to a MOM, or in this case the MQTT server, where data is gathered. Applications then connect to the MQTT server, getting an update whenever there are changes to the data.

The second important feature is the use of a cryptographic security protocol called Transport Layer Security (TLS), which provides communications security over a computer network. TLS aims to provide privacy and data integrity between two communicating computer applications. It is designed to prevent eavesdropping and tampering. By using TLS, MQTT establishes a secure, private connection via a handshake process. Once a connection is made, data is encrypted and transmitted between the client and the server. If the handshake fails, data is not transmitted.

In addition to providing low bandwidth and a high level of security, MQTT has an incredibly useful feature called stateful awareness. While current SCADA implementations purely transmit data from devices, MQTT also sends the device state data about the health of the device or network connection. This is incredibly important for remote locations because it enables operators to determine if network connections are operational or devices are unavailable.

As we dive deeper into the best practices for IIoT, we will discover that stateful awareness is one of the key ingredients to a successful IIoT implementation. Next, let's look into the importance of stateful awareness and how to implement MQTT.



Best Practice #3:
Use MQTT's Built-In Stateful Awareness

With its low-bandwidth publish-subscribe methodology and TLS security, MQTT has proven to be a formidable IIoT communication protocol. Another feature that is critical to your IIoT infrastructure is the stateful awareness that is built into MQTT.

Stateful Awareness is Important for IIoT

Stateful awareness is important for SCADA systems, especially for remote installations.

Knowing the health of the device and the network connection helps to mitigate any downtime and ensures data is being shared with all levels of an organization. By having stateful awareness, data becomes more stable, reliable, and contextual.

Most enterprises still depend on legacy poll-response communication protocols to be able to know the state of the network connection between devices and the SCADA system. Unfortunately, devices must be polled frequently to determine whether or not a network connection is good, which can put a strain on the system.

Stateful Awareness is Built Into MQTT

MQTT has stateful awareness built in and it is the only stateful architecture available. It is designed with a "last will and testament": if the device stops working and falls off the network, then the MQTT server will publish a "death certificate" and the device will be marked as being unable to publish data. On a larger scale, with thousands of devices connected to the MQTT server, it is important to know within seconds the state of each device: whether it is online and ready to publish data or if it is unavailable.

Let's say that you're using the Ignition platform with an MQTT server and an MQTT client. When a device connects to an MQTT server, it registers its state and then it registers its last will and testament. Ignition and the MQTT client will know that the devices are online and will deliver information if and when it changes. In the event of hardware failure or environmental issue, the MQTT server will publish the fact that the device is no longer available. In Ignition, those tags are represented as stale data and the device is marked as unavailable. When the device comes back online, it republishes its birth certificate. The MQTT client knows that the device is available, and Ignition shows the updated tags and the device's availability.

Stateful Awareness Improves Processes

Stateful awareness is a subtle but powerful feature of MQTT. There are many MQTT implementations that are stateless, but for SCADA implementations, stateful awareness is essential. MQTT uses reporting by exception (RBE), which is made possible by

stateful awareness. Data is only sent when there are changes to the state of the device or when data values change, which reduces the amount of useless data taking up bandwidth resources.

Knowing the device state is valuable since it helps to provide context to the data. Operators, especially, can keep track of devices and quickly pinpoint any trouble spots without having to send a technician out to a location to verify an issue. On the organizational level, data can be verified to be up-to-date and accurate.

3 Ways to Implement MQTT

Now that we have established that MQTT is the ideal communications protocol for your IIoT system, our next step is to look at ways to start implementing MQTT. There are three main strategies to transition your SCADA system over to MQTT: converting existing devices to MQTT, enabling existing devices to communicate with MQTT platforms, and embedding MQTT directly onto devices.

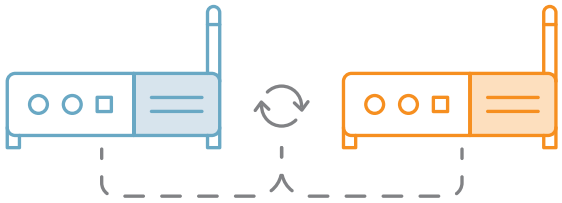
The first strategy is to convert legacy devices to use MQTT. An edge-of-network device is designed to communicate with legacy devices using their native protocol. The edge gateway then connects directly to an MQTT server. The poll-response is moved to the local level, and data is converted to MQTT and published to an MQTT server. This strategy is ideal for current installations using legacy equipment and traditional poll-response protocols.

The second strategy is to enable edge devices to communicate with MQTT platforms using the Sparkplug specification from Cirrus Link Solutions. Cirrus Link provides open-source software, tools, and the Sparkplug reference specification to allow applications, sensors, devices, or gateways to seamlessly integrate with Ignition using the Cirrus Link MQTT modules.

The third strategy, which is appropriate for newer installations, is to use devices with MQTT already embedded into them. Manufacturers have begun to offer devices that have the Sparkplug specification enabled, making them ready to install and connect to your MQTT server and to the rest of your IIoT implementation. In this strategy, these edge-of-network devices do not require a separate edge gateway since the MQTT functionality is already built in.

Now that we have established the importance of stateful awareness and how to implement MQTT, we can turn our attention to edge-of-network devices, which act as a bridge between PLCs and the MQTT server or “broker.” This capability makes them a critical component in the MQTT architecture and IIoT infrastructure.

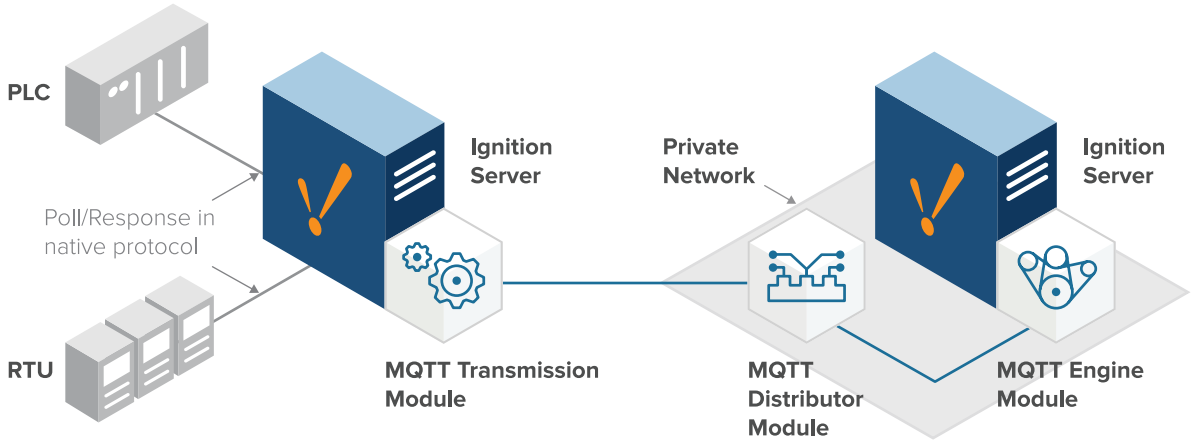
Best Practice #4:
Implement Redundant Edge Gateways



Regardless of whether a location is local or remote, connectivity can pose a major challenge. Local locations tend to rely on hardline connections to transmit data. Remote locations rely on wireless services such as cellular or satellite. In either case, the main communications conduit could potentially fail. As a best practice, especially for mission-critical systems, make sure to integrate failsafes and install redundant edge gateways.

Edge Devices and Gateways

Edge gateways are a type of edge-of-network device. Edge-of-network devices are a key component in MQTT architectures, providing an entry point into an enterprise core network. Edge devices can include routers, routing switches, integrated access devices (IADs), multiplexers, and a variety of metropolitan-area network (MAN) and wide-area network (WAN) devices. An edge gateway can be thought of as a combination of a router, network box, terminal server, and a network arbitrator.



As their name suggests, edge gateways live at the outermost edge of a SCADA system. With an abundance of legacy PLCs and devices still using poll-response communication protocols, edge gateways act as a bridge to connect to these legacy devices, converting them to MQTT, enabling them to communicate to MQTT servers, and allowing enterprise networks to access the data.

Make the Edge Gateways Redundant

Putting in failsafes is a must for your SCADA transition. The data you're collecting and sharing is important and any failures can lead to negative effects on your organization. Because edge gateways are critical, adding a redundant edge gateway is a best practice.

It is also highly recommended that you add redundancy to your IIoT infrastructure as a whole. Make sure there isn't a single point of failure that can cripple your operation. You should add redundancy at every point in the system where data is published.

Make sure you have edge gateways that are able to communicate via cellular and satellite. Set up multiple MQTT servers and use all available channels to make sure data is available at all times.

Always Have Backup Systems

Redundancy is crucial for your IIoT implementation, especially when you have installations in remote geographical areas. Having a failsafe ensures your data is safe and available when it is needed the most. If there are system failures, having backup systems ensures smooth operation and minimizes downtime, which could save your organization thousands or millions of dollars.

The key to a redundant architecture is to take advantage of multiple communication channels. Having a hardline system is always preferable, but having a wireless backup such as cellular or satellite ensures your system has continual coverage to ensure your valuable data is safe and your system keeps running smoothly.

Next, we'll discuss options for the edge devices, MQTT servers, and MQTT clients in your MQTT architecture.

Best Practice #5:
Use MQTT Modules in Your Ignition IIoT

MQTT is an incredible communications protocol that is ideal for your IIoT infrastructure. Yet, you still need an industrial platform that fully takes advantage of MQTT and brings IIoT together. With Cirrus Link's MQTT modules, Ignition becomes the ideal IIoT platform. The Cirrus Link MQTT modules leverage Ignition's rich feature set and superb SQL database capabilities to take existing equipment and systems to create a robust IIoT infrastructure. Depending on your specific needs, be

it an edge gateway, an MQTT server, or enabling MQTT functionality, the Cirrus Link MQTT modules and Ignition have you covered.

Ignition architectures with MQTT are comprised of several elements: edge devices, MQTT servers, and MQTT clients. Each of these elements plays an important role in your ability to take a legacy system and migrate into a cutting-edge SCADA system.

Edge-of-Network Devices

The edge-of-network device is the first important component of the MQTT architecture. Edge devices which include edge gateways (also known as directors) allow you to communicate to legacy devices such as PLCs and RTUs, to collect tag and state data, convert it to MQTT, and publish that data onto an MQTT server. Edge gateways, along with MQTT, allow you to decouple devices from applications, thus improving bandwidth usage.

There are four methods for implementing an edge-of-network device. The first method is to use a dedicated edge gateway to bridge legacy devices to new devices or to an MQTT server.

The second method is to install a brand-new edge device that natively communicates via MQTT. Several manufacturers are now embedding the Sparkplug specification onto their devices, allowing for direct communication with an MQTT server.

The third method is to use the Cirrus Link MQTT Transmission Module on an Ignition server. The MQTT Transmission Module turns the Ignition server into an edge gateway, allowing you to collect and publish edge data to the MQTT server.

The fourth method is to use Ignition Edge MQTT, a limited, lightweight Ignition solution that turns touch panels, client terminals, or virtually any embedded PC or field device into an MQTT-enabled edge gateway.

MQTT Servers

The second piece of the architecture is the MQTT server, often called the broker. This is where the message-oriented middleware (MOM) resides. All edge-of-network devices in the MQTT architecture publish MQTT tag and state data to the MQTT server. The MQTT server then enables MQTT clients to securely connect and subscribe to the device's published data.

Since MQTT is an open standard, there are many companies making their own brand of MQTT servers. For example, there's AWS IoT from Amazon, Azure IoT Hub from Microsoft, and Chariot from Cirrus Link, as well as HiveMQ, CloudMQTT, Red Hat AMQ, and VerneMQ. There are many options to choose from, whether it be a locally hosted or cloud-hosted solution.

As a best practice, use the Cirrus Link MQTT Distributor Module for Ignition. The MQTT Distributor Module is launched by the Ignition Gateway and is a small, self-contained MQTT server. It provides you with a complete MOM solution that is best suited for an on-premise MQTT infrastructure with a limited number of edge-of-network devices. The module provides rapid development and is ideal for situations where communications are restricted or costly. It's designed to simplify and speed up the process of getting a complete MQTT infrastructure going. It's also very effective at increasing data throughput for high-performance plant-floor solutions.

MQTT Clients

The third piece of the MQTT architecture is the MQTT client, which connects to the MQTT server and subscribes to one or more topics of information, bringing that data right into an application.

While there are many MQTT client tools available, it is recommended to use the Cirrus Link MQTT Engine Module for Ignition. The MQTT Engine Module is the key component that enables Ignition to act as a native MQTT citizen.

The MQTT Engine allows the Ignition platform to communicate bidirectionally with MQTT-enabled edge-of-network devices via the MQTT server, which can be sent securely all the way down to the device. It takes data from the MQTT server and injects it into industrial SCADA applications.

Now that we have covered all the available options, resources, and best practices, it is time to look at how to bring everything together. In the last section, we'll take a look at the best migration strategy to take when implementing an IIoT infrastructure.

Best Practice #6: Follow the Optimal IIoT Migration Strategy

Migrating to a new IIoT infrastructure takes time and careful planning. As we discussed in the first best practice, you must approach the migration transition slowly. There are many pieces involved with an IIoT infrastructure and when you factor in the added complexity of a legacy system, you need a slow transition to ensure that downtime is minimized.

Many organizations face a Catch-22 when implementing a SCADA infrastructure upgrade. Current SCADA solutions still use the poll-response protocol drivers that are directly connected to field devices over a communications channel or a TCP/IP network. Because of this, they cannot replace or upgrade the poll-response protocol on the SCADA host until the new protocol is in the field and, vice versa, they cannot upgrade devices until they have a new protocol on the SCADA host.

The Best Way to Migrate Your System

There is a proven four-step migration strategy that uses Ignition, the MQTT Engine Module, and an edge gateway. The four steps are installing an edge gateway, starting the poll-response with Ignition, enabling MQTT local masters, and switching over to the pure MQTT solution.

Step 1: Install and set up an edge gateway, such as an Elecsys Director. The edge gateway acts as a TCP/IP cellular, VSAT, or Ethernet IP endpoint. In this step, you are setting up the edge gateway to communicate with PLCs or edge devices using the poll-response protocol. The edge gateway then connects to an MQTT server using a TCP/IP link and sends collected data via the MQTT protocol.

Step 2: Start the conventional poll-response with Ignition by connecting to the internal terminal server and keeping poll-response going. You will enable both the serial and Ethernet connections between Ignition and the field devices using the edge gateway.

Step 3: Enable MQTT local masters. Once the conventional poll-response protocols have been enabled, legacy SCADA users can start to see the conventional use of Ignition. You now have a parallel communication system where you can compare values coming in from the conventional poll-response and publish-subscribe of MQTT.

Step 4: You are ready to switch over to your pure MQTT solution. Once the organization is happy with the final, pure MQTT infrastructure, you can remove the OPC poll-response components. This will greatly simplify the network topology and leave you with a clean and pure MQTT solution. Furthermore, you now have a plug-and-play SCADA system that is easily scalable and can quickly grow with your organization.

Best Practices Recap

To recap the best practices for a successful IIoT implementation, we recommend:

- Planning your strategy and adding to your infrastructure in pieces, giving yourself opportunities to test and make sure everything is working.
- Choosing MQTT as your IIoT messaging protocol since its feature set is ideal for an IIoT infrastructure.
- Leveraging stateful awareness to help maintain the health of your IIoT infrastructure.
- Employing a redundant strategy throughout your solution.
- Considering Ignition with the Cirrus Link MQTT Engine, Transmission, and Distributor Modules as your IIoT solution.
- Finally, following the optimal migration strategy to ensure a smooth transition to a new IIoT infrastructure.

The best practices discussed in this paper are proven methods for success. Even if you have a legacy system, the solutions we have offered reduce friction and give your organization the resources to move an existing installation into a state-of-the-art IIoT and SCADA solution.

To learn more about how Ignition delivers the industry's best IIoT solution, visit: ***inductiveautomation.com***