VALTORIS.

APPLICATION NOTE

OT-to-IT Network Topology Blueprint: Integrating Serial Modbus into Ignition SCADA via MQTT

Document No:	AN-2026-06
Product Series:	Valtoris VT-DTU Series / Edge Gateways
Target Platforms:	Ignition SCADA (MQTT Engine), AWS IoT, EMQX
Date:	June 2026 (Rev. 1.0)

1. Executive Summary

Inductive Automation's Ignition platform provides robust OT/IT convergence. However, integrating legacy RS485 Modbus RTU devices across wide-area networks (WAN) or cellular connections presents critical stability risks. When utilizing standard "Serial-to-Ethernet" transparent forwarders, the Ignition OPC UA server must actively poll field devices over TCP.

This legacy architecture introduces severe latency, bandwidth inefficiency, and continuous timeout errors (such as `16#80C8`) whenever network jitter occurs. This Application Note outlines the industrial best practice: deploying an Edge-Decoupled MQTT Architecture to shift the polling burden to local silicon and stream clean, pre-parsed JSON payloads directly into Ignition.

2. The Challenge of Legacy Serial Polling

Before adopting MQTT, engineers typically bridged RS485 to Ethernet. This forces the central SCADA to act as the Modbus Master over the internet:

- **Latency Sensitivity:** Modbus RTU is highly sensitive to timing. A 200ms cellular delay causes the SCADA to assume the device is offline.
- **Bandwidth Exhaustion:** Continuous polling requests (e.g., polling every 1 second) flood the network, even when field data values have not changed.

- **Scalability Bottlenecks:** A central server polling hundreds of remote serial chains will quickly experience thread exhaustion and CPU overload.

3. The Edge-Decoupled MQTT Architecture

To achieve true enterprise scalability, the polling cycle must be physically decoupled from the SCADA server. This is achieved by deploying an **Industrial Edge Gateway (DTU)** at the field layer.

The Valtoris Decoupled Topology:

- **Local Acquisition:** The Edge Gateway polls the local RS485 bus via Modbus RTU (e.g., 9600, 8N1). Because this occurs locally over copper wire, timeout errors are virtually eliminated.
- **Edge Normalization:** The Gateway translates hexadecimal registers into float/integer values natively. It handles byte-swapping and parity logic, eliminating the need for complex Python scripts within Ignition.
- **Event-Driven Push:** The Gateway packages the normalized data into a standard MQTT JSON (or Sparkplug B) payload and publishes it to Ignition's MQTT Distributor only upon change or at set intervals.

4. Configuration Phase A (Edge Gateway)

Instead of sending raw hex code, configure the Edge Gateway to act as the Modbus Master and translate the register map into readable JSON keys.

Step 1: Data Aggregation & Bandwidth Optimization

Enable the **"Report on Exception"** or **"Deadband"** feature on the Valtoris DTU. Configure the gateway to only publish an MQTT message when a value changes by >1%, drastically reducing cellular data usage.

Step 2: Payload Structure Definition

The Edge Gateway must be configured to output a flattened, structured JSON payload. Below is the standard template expected by modern IoT brokers:

```
{
  "timestamp": "2026-06-15T08:30:00Z",
  "gateway_id": "VT-DTU-Substation_Alpha",
  "devices": {
    "Chiller_Meter_1": {
      "Voltage_A": 230.5,
      "ActivePower_kW": 45.2,
      "Status": 1
    }
  }
}
```


Note: Because the DTU handles the IEEE 754 Float32 byte-swapping natively, the payload delivers ready-to-use decimal values (e.g., 230.5) instead of raw Modbus hex.

5. Configuration Phase B (Ignition SCADA)

With the data pre-formatted as JSON by the Edge Gateway, integration into Ignition requires zero coding or manual tag mapping.

Step 1: Install Required Modules

Ensure Cirrus Link's `MQTT Distributor` (which acts as the local MQTT Broker) and `MQTT Engine` (which acts as the MQTT Client) modules are installed and activated on your Ignition Gateway.

Step 2: Namespace Subscription

- Navigate to the **MQTT Engine Settings** in the Ignition Gateway webpage.
- Go to **Custom Namespaces**.
- Subscribe to the Valtoris DTU's publishing topic (e.g., `valtoris/telemetry/site_alpha/#`).

Step 3: Auto-Discovery & Tag Generation

Once the Edge Gateway publishes its first payload, Ignition's MQTT Engine will automatically parse the JSON keys. It dynamically generates UDT (User Defined Type) tags for `Voltage_A` and `ActivePower_kW` directly in the Tag Browser. The data is now live and ready for Perspective dashboards.

6. Hardware Selection Criteria

When selecting an Edge Gateway or DTU for Ignition integration, ensure the hardware meets the following strict industrial specifications:

- **Galvanic Isolation:** Minimum 1.5kV to 2500V opto-isolation on RS485 ports to protect against VFD common-mode noise and ground loops.
- **Protocol Support:** Native Modbus RTU master capabilities, paired with MQTT (TLS 1.2/1.3 encryption).
- **SRAM Caching:** Built-in memory cache to store data during cellular network drops, ensuring zero data loss upon reconnection.

*For enterprise-grade edge hardware that natively supports these specifications out-of-the-box, explore the **Valtoris VT-DTU Series** at **valtoris.com**.*