# MQTT Client for kuksa.val and AWS IoT: Project Specification

## Introduction

This project aims to develop an MQTT client that interfaces with the kuksa.val platform for accessing vehicle signals. The client will provide compatibility with AWS IoT leverage current standards for vehicle communication. Security will be enforced through mTLS authentication.

## Definition of Done

### Functional Requirements

1. **MQTT Client Implementation**: The client must successfully interface with kuksa.val to access vehicle signals.
2. **AWS IoT Compatibility**: A working reference implementation for AWS IoT is required, including device onboarding and data routing to Amazon Timestream DB.
3. **Compliance**: The client must adhere to current VSS specifications.
4. **Vehicle Identification**: The client must be able to resolve vehicle identification numbers or other unique identifiers.
5. **Protobuf**: Payload definition for MQTT is defined.
6. **Configurability**: The client should be adaptable to work with any MQTTv5 broker.
7. **AWS IoT Demo**: A working demo with AWS IoT device onboarding, message routing to Amazon Timestream DB, and vehicle signal visualizations through Amazon Managed Grafana.

### Non-Functional Requirements

1. **Documentation**: Comprehensive documentation must exist, detailing the architecture, setup, and usage instructions, including steps for the AWS IoT demo.
2. **Unit Tests**: All critical functionalities, including the AWS IoT demo, must be covered by unit tests.
3. **Peer Review**: The code must be reviewed by at least two engineers and be approved.
4. **Security**: mTLS should be used as the primary method for secure authentication.

## Requirements

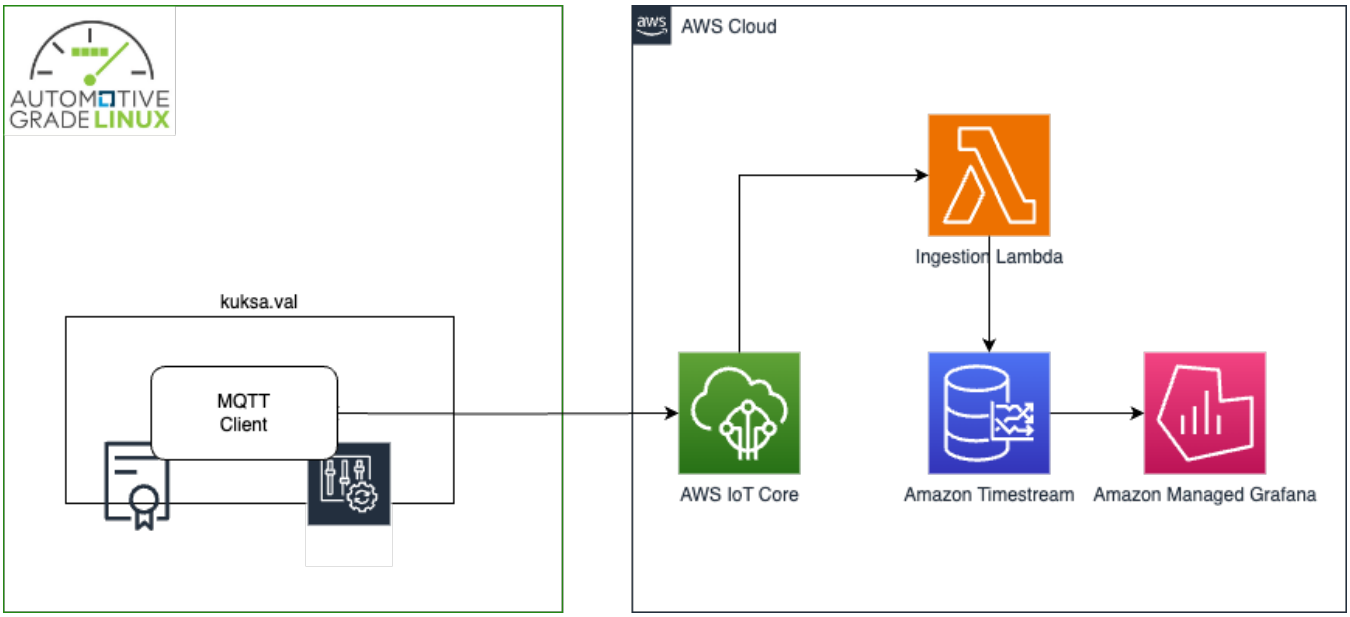
### Functional Requirements

1. **Language and Libraries**: The client implementation should be done in C/C++, Python is an option as well.
2. **MQTT Protocol Version**: The client must be compatible with MQTTv5.
3. **Cloud Architecture**: Architecture details for interfacing with AWS IoT, including components for device onboarding and message routing to Amazon Timestream DB.
4. **Data Schema**: Define a schema for vehicle signals based on VSS
5. **Protobuf Specifications**: Focusing on signal updates.
6. **Configuration File**: Design a user-configurable file for broker details and other settings.

### Non-Functional Requirements

1. **Unit Testing Framework**: Identify the framework and libraries to be used for unit testing.
2. **Documentation Standards**: Establish standards and templates for documenting the code and architecture.
3. **Code Review Process**: Outline the process and criteria for peer code reviews.
4. **Security Protocols**: Specify the mTLS version and cipher suites that will be supported.

### High level architecture




## Next Steps

1. **Migrate this document to Confluence**
2. **Task Break Down:** Start breaking down the work needed to deliver the mentioned DoD
    a. Interfaces definition
    b. Cloud breakdown
3. **Task Allocation**: Assign specific components and tasks to team members, including tasks related to the AWS IoT demo.
4. **Development Timeline**: Establish milestones and deadlines, ensuring that the demo is ready for a specified milestone.
    a. Embedded World Demo as a target
        i. Start in January
        ii. Targeting March for vehicle part
        iii. End by April
5. **Community Outreach**: Strategies for involving the AGL community in project feedback and contributions.

Monday, October 30

## Task Breakdown

### Device Part:

- [x] Determine the Signals for Demonstration

- **Description**: Identify which CAN messages are available and decide which vehicle signals should be sent to the cloud for monitoring.
- **Owner**: Nenad
- **Timeline**: 13th of November
- **Dependencies**: None
- **Resources**: CAN message logs, kuksa.val documentation
    - Nenad will get the list of signals from the the current FW demo and ask around about what are signals are used

- [x] Signal Forwarding Logic

- **Description**: Decide the frequency or event triggers for forwarding the selected vehicle signals.
- **Owner**: Scott
- **Timeline**: [mid Jan - end Jan]
- **Dependencies**: Determine the Signals for Demonstration
- **Resources**: Engineering specifications
    - Decided to go with time based and trigger implementation located in a local config file
    - Write an example of the file for a specific usecase

- [x] Proto File Generation

- **Description**: Generate a proto file that defines the messaging format
- **Owner**: [Name/Role]
- **Timeline**: [mid Jan - end Jan]
- **Dependencies**: Signal Forwarding Logic
- **Resources**: Protocol Buffers info
    - Define the timestamp

        - identifier, do we need it to be part of the payload, or an MQTT topic, or we can simply use the MQTT client ID
        - This is a prerequisite for building out the cloud part of ingestion

- [ ] Vehicle Onboarding

  - **Description**: Design and implement the mechanism for vehicle onboarding, including certificate generation and client ID definition.
  - **Owner**: Scott
  - **Timeline**: [end Jan - mid Feb]
  - **Dependencies**: None
  - **Resources**: AWS IoT documentation
    - How do we get a vehicle identifier (VIN)?
      - Populate values in VSS perhaps?
      - This should part of that static config
        - This should come from a TPM in a real world scenario

- [ ] MQTT Device Client Development

  - **Description**: Develop the MQTT client to interface with kuksa.val and AWS IoT based on previous task findings.
  - **Owner**: Scott
  - **Timeline**: [end Feb - mid March]
  - **Dependencies**: Create Proto File, Vehicle Onboarding
  - **Resources**: MQTTv5 specifications, AWS SDK, other MQTT Clients

### Cloud Components:

- [x] IaaC Setup

  - **Description**: Set up Infrastructure as Code for provisioning and managing cloud resources.
  - **Owner**: Nenad
  - **Timeline**: [Start Date - End Date]
  - **Dependencies**: None
  - **Resources**: AWS CloudFormation or CDK

- [x] Image Parser Component

  - **Description**: Develop a component to parse the vehicle signal images for cloud ingestion.
  - **Owner**: Nenad
  - **Timeline**: [23rd of Feb - 28th of Feb]
  - **Dependencies**: IaaC Setup, Device part: Proto File generation
  - **Resources**: AWS Lambda

- [x] Time Stream Injector Component

  - **Description**: Implement a component to insert the parsed signals into the Amazon Time Stream database.
  - **Owner**: Nenad
  - **Timeline**: [28th of Feb - 5th of March]
  - **Dependencies**: Image Parser Component
  - **Resources**: Amazon Time Stream DB documentation

- [x] Time Stream Data Modeling

  - **Description**: Model the Time Stream database to accommodate the vehicle signals.
  - **Owner**: Nenad
  - **Timeline**: [5th of March - 14th of March]
  - **Dependencies**: Time Stream Injector Component
  - **Resources**: Data modeling tools

- [ ] Grafana Setup for Data Visualization

  - **Description**: Set up Grafana to create dashboards for visualizing vehicle signals.
  - **Owner**: Nenad
  - **Timeline**: [ 14th of March - 20th of March]
  - **Dependencies**: Time Stream Data Modeling
  - **Resources**: Amazon Managed Grafana documentation

- [ ] Integration Work

  - **Description**: Integration of device MQTT client and cloud components .
  - **Owner**: Nenad/Scott
  - **Timeline**: [start March - end March]
  - **Dependencies**: All other task
  - **Resources**: None