Detailed Technical Response to Mr. HS Cho’s Questions

# Q1: I'm curious about the process of users being registered in VTN. during AHRI certification test

The AHRI test lab sets up an OpenADR 2.0b VTN server. Our SmartThings Cloud Pseudo-VEN needs credentials before testing. These can be a client certificate or username/password. Registration uses EiRegisterParty service from OpenADR 2.0b.

The registration follows specific steps.

1. The Pseudo-VEN sends a HTTPS POST to the VTN endpoint.
2. This POST includes venID, venName, and profile B support.
3. Authentication happens via TLS with certificates or HTTP Basic Auth.
4. The VTN then confirms registration. It may assign a new venID if needed.

AHRI labs require specific XML data formats. Message payloads must include vendorID (Samsung), deviceClass (AC), and resourceID fields.

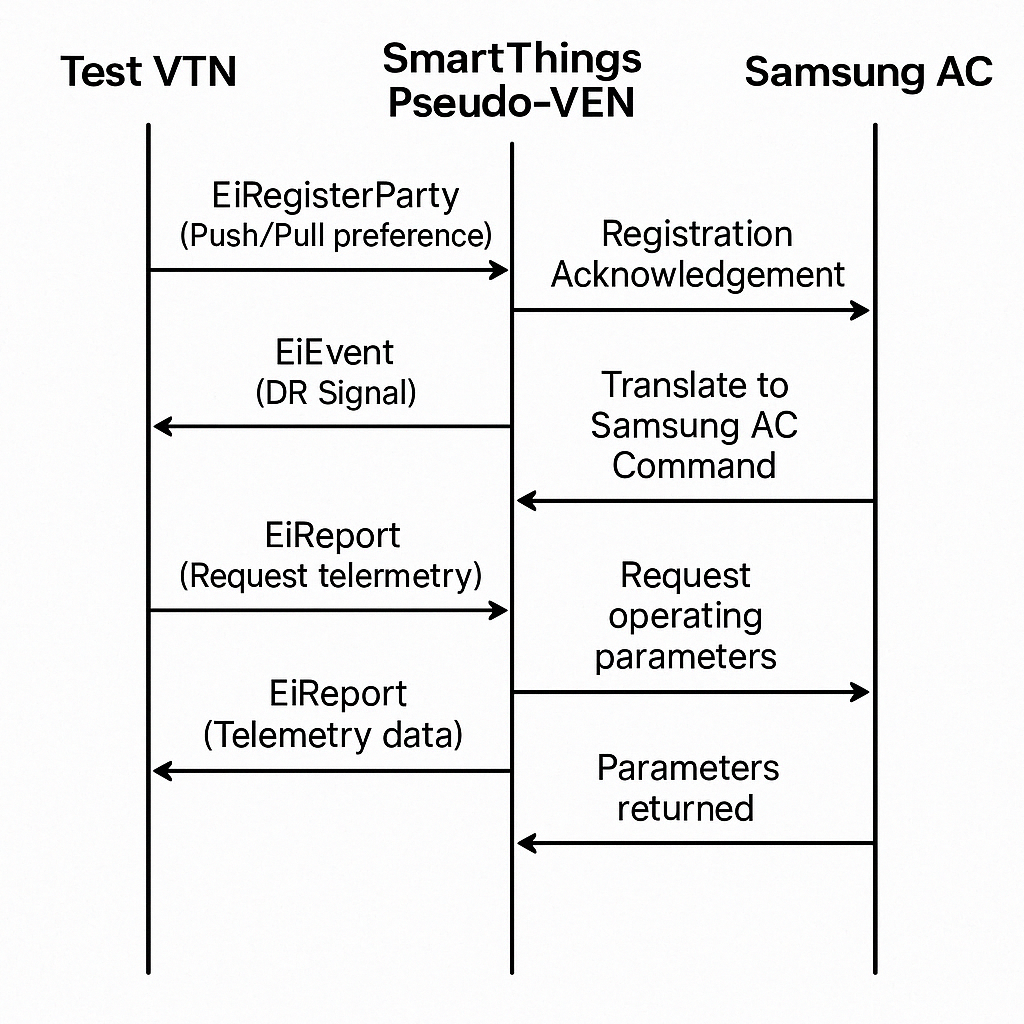
Each OpenADR message must have a unique messageID in UUID format. The EiRegisterParty payload must specify reportCapability and optCapability as "true".

Our system must log all registration messages. Store both request and response XMLs with timestamps. Include TLS handshake logs. Use these logs as certification evidence. The registration must complete within 30 seconds for certification.

# Q2: What is Pseudo-VEN? What is the relation between Pseudo-VEN and SmartThings Energy(pdr-data)?

A Pseudo-VEN is a cloud proxy for OpenADR signals. It implements OpenADR 2.0b client protocol. However, it doesn't control loads directly. Instead, it converts OpenADR signals to Samsung commands.  
  
The SmartThings Cloud hosts the Pseudo-VEN code. We can run as microservices in Kubernetes pods. The OpenADR parser service (which we will need to develop) will handle all XML processing.

The event mapper service translates DR signals to device commands. The scheduler service manages timing of all events.

The pdr-data service in SmartThings Energy stores all DR events. It uses a PostgreSQL database for persistence. Each event record contains eventID, startTime, endTime, and signalType fields. The system assigns priority levels from 1-5 based on event criticality.

The device control flow works in precise steps. First, pdr-data receives the translated event. Then it creates device-specific command objects. These objects contain the commandType and parameters for each AC. The command delivery service sends MQTT messages to devices. Each message uses QoS level 1 to ensure delivery.

# Q3. The particular air-conditioned device can be the testing device for AHRI certification.

Yes, we can start by selecting one specific AC model for testing. AHRI requires complete device details. We will record the model number, serial number, BTU rating, and firmware version. Register this AC in SmartThings with a dedicated test account. Create a specific venID like "samsung-ac-test-001" for this device.  
  
*Approach 1: Direct VEN Approach*

Configure the AC with special test firmware. This firmware must implement four DR modes.  
  
Mode 1 – Compressor Shutdown mode  
Mode 2 - Temperature Setpoint Adjustment Mode:  
Mode 3 – Duty Cycling Mode  
Mode 4 – Energy Efficiency Mode  
  
AHRI 1380 does not compulsorily require all 4 DR modes. The standard requires Samsung to support at least one of these Demand Response capabilities, not all of them.  
  
We will ignore the compressor turn off modes (Modes 1 and 3 – as we are aware for residential ACs, turning off compressor and turning back on is not done). Mode 2 increases setpoint by exactly 2°F (1.1°C). Mode 4 activates Samsung's "Eco" energy-saving mode.

Enable extended logging on the test device. Log every state change with millisecond timestamps. Record power consumption at 5-second intervals. Track temperature setpoint adjustments. These logs must be exportable in CSV format for certification.

The test lab environment controls ambient conditions. They maintain room temperature at 80°F (26.7°C). They set humidity at 50% ±5%. Power monitoring equipment measures load at 100ms intervals. Response time measurement starts when the VTN sends the signal and ends when power usage changes.  
  
*Approach 2: Pseudo-VEN Approach*  
  
In this approach, no firmware modifications are needed. This is a key advantage of the Pseudo-VEN approach.

The SmartThings Pseudo-VEN translates OpenADR signals to existing AC commands. We can test directly with Mode 2, it uses the standard "setpoint\_increase" command with a 2°F parameter. For Mode 4, it sends the existing "eco\_mode" command. No firmware changes are required on the device.

Enable extended logging in the SmartThings Cloud, not on the device. The Pseudo-VEN logs every command sent with millisecond timestamps. The SmartThings Energy platform records device state changes. It logs power consumption at 5-second intervals through existing power monitoring. It tracks compressor status through standard device telemetry. All logs are stored in the cloud and exportable in CSV format for certification.

The test lab environment controls ambient conditions. They maintain room temperature at 80°F (26.7°C). They set humidity at 50% ±5%. Power monitoring equipment measures load at 100ms intervals. Response time measurement starts when the VTN sends the signal and ends when power usage changes. This approach fully leverages existing AC capabilities without custom firmware development.

4. ***I guess that there is some VTN for testing purpose for AHRI certification test?***

AHRI labs use certified OpenADR 2.0b VTN software, available here:  
  
<https://www.openadr.org/epri-certified-open-source-code>   
  
This software passes OpenADR Alliance certification tests. It supports all OpenADR 2.0b profiles and services. It generates all signal types required by AHRI 1380.

The test VTN uses specific signal patterns. It sends SIMPLE signals with "oadrDistributeEvent" payloads. These contain "oadrEvent" elements with unique eventIDs. MODERATE signals include "oadrSignal" elements with temperature offset values. COMPLEX signals contain detailed "oadrSignalPayload" elements with duty cycling parameters.

VTN communication uses synchronous HTTP for EiRegisterParty. It uses asynchronous PUSH or PULL for events. PUSH mode delivers events directly to our VEN endpoint. PULL mode requires our VEN to request events every 60 seconds. Our Pseudo-VEN must support both modes.

*SRI-B can help to build a complete VTN simulator before certification.*   
  
We can start by using OpenADR 2.0b open-source libraries like OpenLEADR. Configure it with the same event patterns as AHRI test VTNs. Test with various payload sizes and timing conditions. Validate responses against the OpenADR 2.0b schema.

# Q5.In what way should the connection between the two be specific?

The connection uses strict security protocols. Implement HTTPS with TLS 1.2 or higher. Configure cipher suites to TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384. Set certificate key length to 2048 bits minimum. Configure perfect forward secrecy (PFS) support.

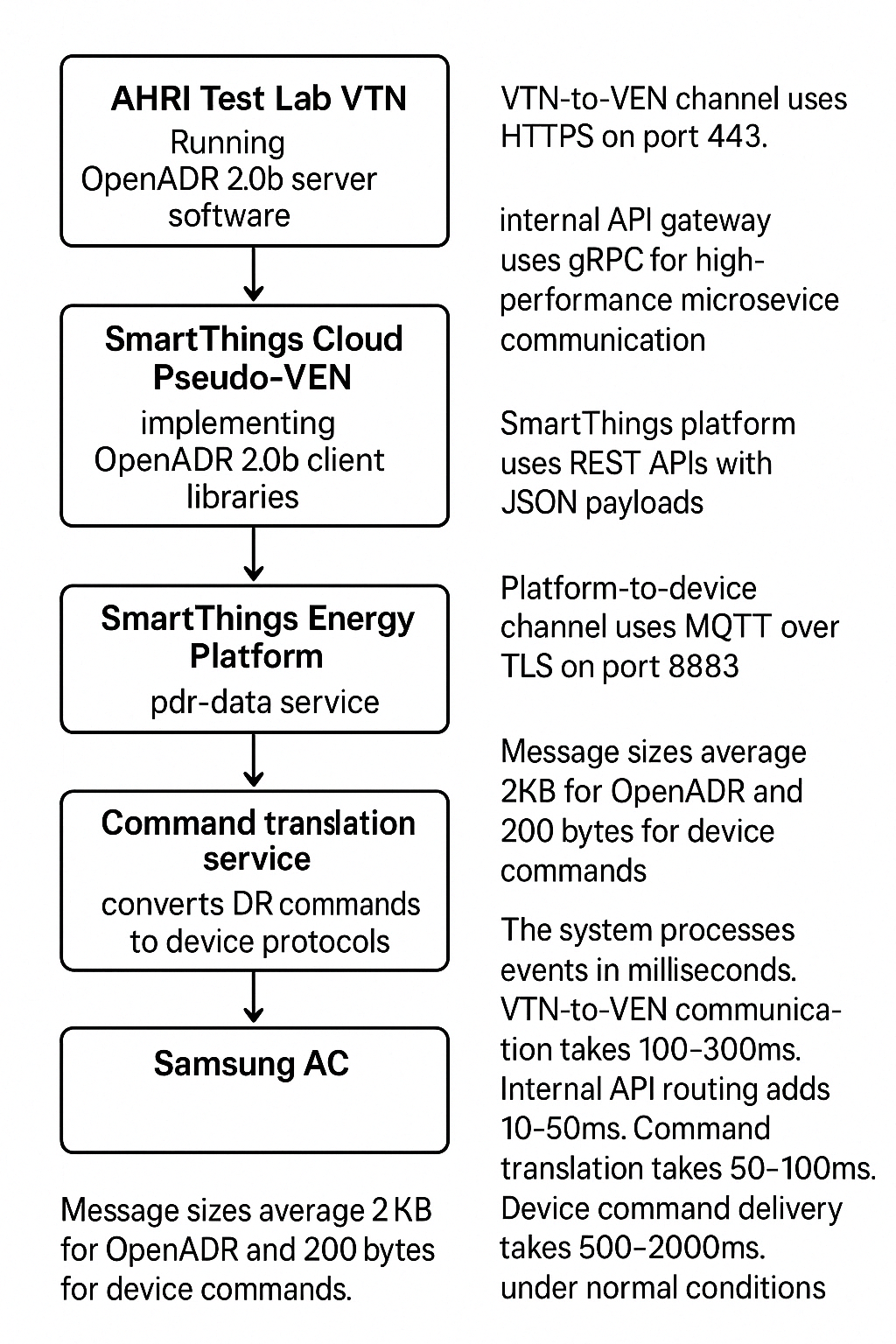
Authentication uses X.509 certificates. The VEN certificate needs organizationName field set to "Samsung Electronics". The commonName field must match our registered venID. Request certificates with 1-year validity periods. Implement certificate pinning to prevent MITM attacks.

The XML payloads follow specific formats. Use XML schema validation against "oadr\_20b.xsd". All messages require an "eiResponse" element with responseCode field. Success responses use code "200". Error responses use codes "400"-"500". Include responseDescription for all error codes.

The complete signal flow works step by step.

1. First, the VTN initiates an oadrDistributeEvent. Our Pseudo-VEN validates the signature using the VTN's public key.
2. It sends an oadrCreatedEvent response within 5 seconds.
3. The Pseudo-VEN translates this to Samsung commands using MQTT topic "samsung/device/{deviceId}/dr/command".
4. The device acknowledges on topic "samsung/device/{deviceId}/dr/ack". For reporting, the VTN sends oadrRequestReport.
5. Our Pseudo-VEN requests telemetry from the AC via HTTPS GET to "{deviceId}/status".
6. It formats this data in an oadrUpdateReport response.

# Q6. What is the End-to-End Architecture for AHRI Certification?

The system architecture has **six distinct layers.**

1. Layer 1 is the AHRI Test Lab VTN running OpenADR 2.0b server software.
2. Layer 2 is the SmartThings Cloud Pseudo-VEN implementing OpenADR 2.0b client libraries.
3. Layer 3 is the internal API gateway that handles routing and rate limiting.
4. Layer 4 is the SmartThings Energy Platform with the pdr-data service.
5. Layer 5 is the command translation service that converts DR commands to device protocols.
6. Layer 6 is the Samsung AC itself.

Data flows through specific channels. The VTN-to-VEN channel uses HTTPS on port 443. The internal API gateway uses gRPC for high-performance microservice communication. The SmartThings platform uses REST APIs with JSON payloads. The platform-to-device channel uses MQTT over TLS on port 8883. Message sizes average 2KB for OpenADR and 200 bytes for device commands.

The system processes events in milliseconds. VTN-to-VEN communication takes 100-300ms. Internal API routing adds 10-50ms. Command translation takes 50-100ms. Device command delivery takes 500-2000ms. Total system latency ranges from 0.66-2.45 seconds under normal conditions.

Failover mechanisms ensure reliability. We also will have to implement active-passive redundancy for the Pseudo-VEN, such as using database replication for pdr-data with 5-second RPO.

Implement exponential backoff for retry logic. Set maximum retry attempts to 5 with a 30-second timeout.

# Key Technical Implementation Requirements

To get AHRI certification, response time optimization is critical.   
  
So, throughout the process, we will measure end-to-end latency at each system boundary.   
  
Some other things we can do for implementation:

1. Implement connection pooling for all HTTP requests.
2. Set TCP keepalive intervals to 60 seconds.
3. Use WebSockets for device connections when available.
4. Maintain connection quality metrics.
5. Target average response time under 5 minutes.
6. Track 95th percentile response times.
7. Create automated alerts for responses exceeding 10 minutes.

***Event handling requires robust logic.***   
  
Implement an event priority system based on OpenADR "oadrPriority" values. Store currently active events in an in-memory cache with Redis. Create conflict resolution rules for overlapping events. Handle communication failures with store-and-forward queuing. Implement device offline detection with 120-second timeouts. Create event replay capability for missed events.

***Security implementation must be comprehensive.***   
  
Use mutual TLS (mTLS) for all service-to-service communication. Implement JWTs with 15-minute expiration for internal API authorization. Store all secrets in a secure vault like HashiCorp Vault. Use AES-256 encryption for data at rest. Implement rate limiting at 100 requests per minute per endpoint. Deploy Web Application Firewall (WAF) rules to block XML injection attacks. Conduct penetration testing before certification.