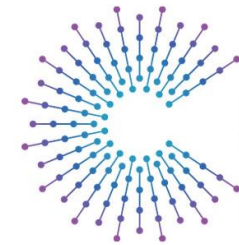




Leveraging Sparkplug for efficiently transmitting VSS data between the vehicle and the cloud

Dr. James J. Hunt | CTO & CEO aicas | September 25, 2024



COVESA

Accelerating the future of connected vehicles

Evolution of Data Collection

Static Aquisition

- Predefined Signals
- Limited control



Rules w/ Fixed Operations

- Downloadable Rules
- Predefined Signals
- Predefined Operations
- Frequency Control
- Event Triggers



Dynamic Network & Nodes

- Network of Nodes
- Dynamically Updatable
- Dynamically Defined Nodes
- Based on Application Framework
- Eases Merging Multiple Data Acquisition Plans
- VSS selfawareness

Vehicle Data Collection Requirements



- Efficient Transport
- Flexible Interconnection of Vehicles and Cloud Services (Publish-Subscribe Messaging)
- Topic and Message Structure Compatible with VSS
- Address More than One Device in a Vehicle
- Fast Fail-Over
- Interoperable
- Selfaware Vehicles
- Flexibility (adaptable to current and upcoming business cases)
- Minimal Reinvention (avoid Not-Invented-Here Syndrome)
- Secure communication

Architecture

- Data Model—VSS
- Data Conversion—vehicle specific formats to a standard format (Data Conversion Manifest)
- Data On-Board Processing—synthetic, composite, and predictive signals (Signal Apps)
- Data Collection—what should be collected when (Data Acquisition Plan)
- Data Transmission—standard means of bringing data efficiently and securely to the cloud

Software Defined Vehicle Concerns

- Combining Robustness and Dynamic Update
- Complete Data Types: Quantities vs. Raw Numbers
- Software Selfawareness
- Dynamic Test and Deployment

Vehicle Data Collection Process

Conversion

- Convert native data to vehicle independent signals
- Partially covered by VSS
- No filtering
 - Only CAN & AUTOSAR signals.

Signal Conversion Manifest

Preprocessing

- Compose signals not present natively.
- Compress & Store data
- Local subsystem modeling.

Software Bundles

Selection

- Decide what signals to send when
- Uses both converted signals and preprocessed data
- Describable as a network

Data Acquisition Plan

Transmission

- Connect to server
- Transmit requested data
- Handle all failure modes such as network disconnect and server failure

Sparkplug


Industry-Proven IoT Protocol over MQTT



- +** Interoperable **>** Seamless communication between devices from different vendors.
- +** Efficient **>** Optimized data transmission for low-bandwidth networks.
- +** Scalable **>** Supports large deployments without overwhelming the system.
- +** Time-saving **>** Standardized communication simplifies development and custom coding.
- +** Reliable **>** Ensure data integrity and system operation regardless of network reliability.

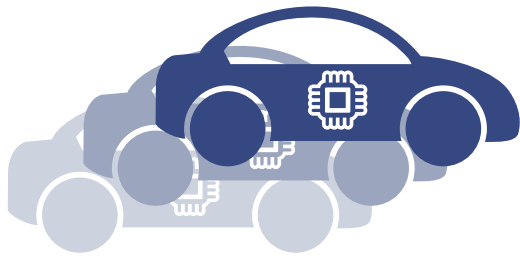
- **Broker based system (MQTT):** a level of indirection provides loose coupling, which facilitates scalability and flexibility in deployment and maintenance (unlike rest-based OPC/UA).
- **Continuous Session Awareness:** provides a standard for managing the session state of all connected components, which facilitates coordination and synchronization.
- **Persistent connections:** all participants are connected continuously by default.
- **Protocol build-in failover:** avoids single point of failure by providing timely reaction to transmission and broker faults by rolling over to another defined broker.
- **Payload definition and interoperability:** uses consistently interpreted data types across the ecosystem. When a new device connects, the initial message exchange defines what to expect.
- **Report by Exception:** data and failure message sent only when data changes
- **Build-in data encoding and compression:** based on Protobuf, provides for interoperability and bandwidth optimization.
- **Defined topic naming:** Aligns well with VSS.
- **Standardized:** ISO/IEC 20237:2023

Sparkplug for VSS Architecture



Sparkplug[®] (ProtoBuf)

Devices in Vehicles



MQTT Broker



Cloud Services



- Vehicle Control
- Historian
- Analytics

TCP/IP

namespace/group_id/message_type/edge_node_id/[device_id]

- *namespace*—The protocol version identifier, e.g., "spBv1.0"
This could be defined for VSS, e.g., "spvssv3.0"
- *group_id*—This would be reserved for an OEM fleet identifier.
- ***edge_node_id***—This would be an OEM defined identifier for a vehicle.
- *device_id*—Optional device identifier, e.g., TPU (telematics Processing Unit)

Payload for data command (metric) and can be mapped directly to VSS.

Sparkplug Message Types

- NBIRTH—Birth certificate for MQTT EoN nodes.
- NDEATH—Death certificate for MQTT EoN nodes.
- DBIRTH—Birth certificate for Devices.
- DDEATH—Death certificate for Devices.
- NDATA—Node data message.
- DDATA—Device data message.
- NCMD—Node command message.
- DCMD—Device command message.
- STATE—Critical application state message.

JSON Serialization Example of NBIRTH Message

```
{
  "timestamp": 1486144502122,
  "metrics": [{
    "name": "bdSeq",
    "timestamp": 1486144502122,
    "dataType": "Int64",
    "value": 0
  }, {
    "name": "Vehicle/Cabin/Tempature",
    "timestamp": 1486144502122,
    "dataType": "Tempature",
    "value": 3000
  }, {
    "name": "Vehicle/Cabin/Humidity",
    "timestamp": 1486144502122,
    "dataType": "Humidity",
    "value": true
  }],
  "seq": 0
}
```

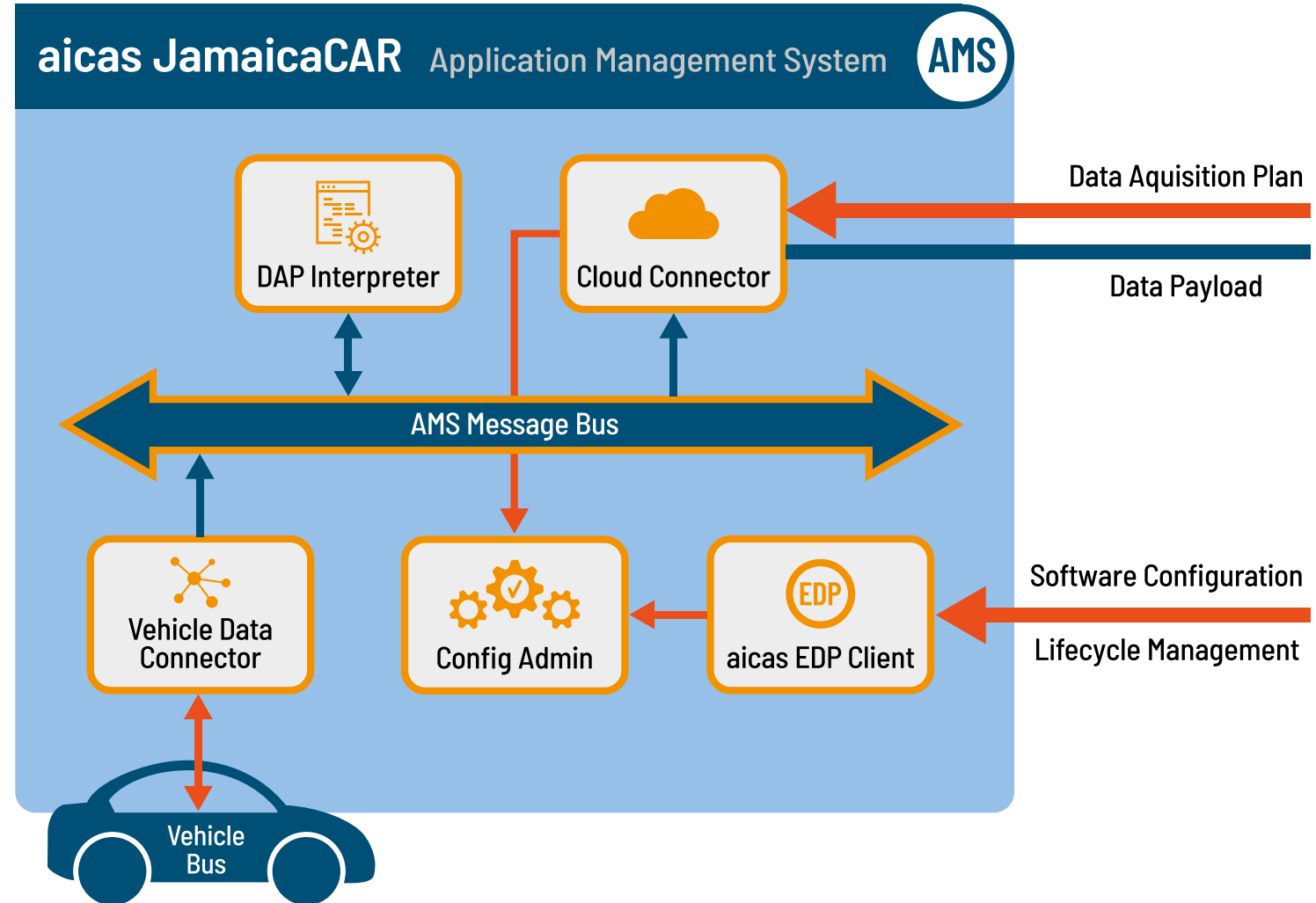
JSON Serialization Example of NDATA Message



```
{
  "timestamp": 1486144502122,
  "metrics": [{
    "name": "Vehicle/Cabin/Temperature",
    "timestamp": 1486144502122,
    "dataType": "Celsius",
    "value": 29.2
  }, {
    "name": "Vehicle/Cabin/Humidity",
    "timestamp": 1486144502122,
    "dataType": "%",
    "value": 55.9
  }],
  "seq": 0
}
```

Integrating a Data Acquisition Plan

1. Define a Data Acquisition Plan as network description
2. Add a Data Acquisition Plan to the rebirth command
3. Interpret a rebirth with a Data Acquisition Plan to update the data dictionary and the internal Signal Conversion Manifest.

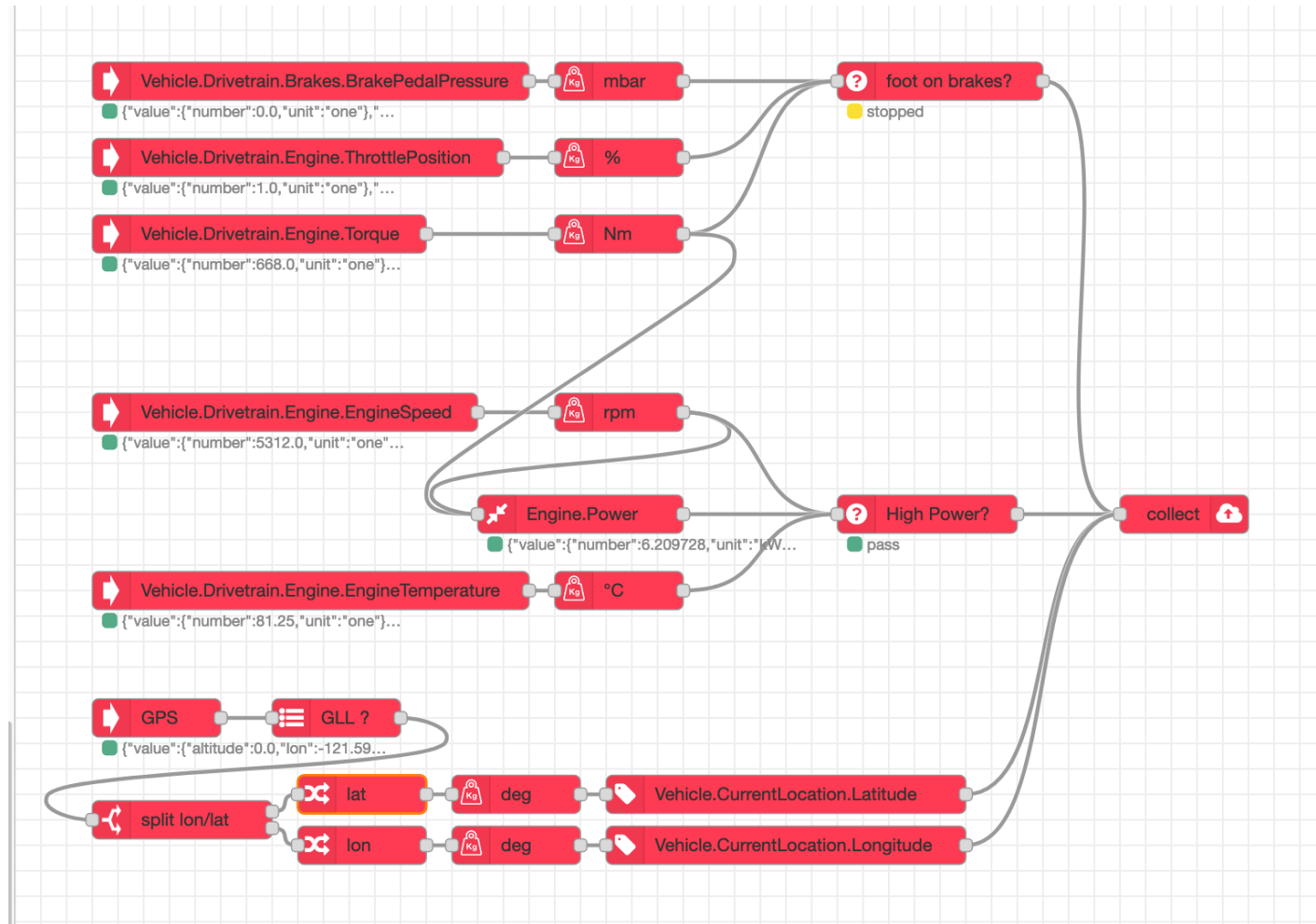


Data Acquisition Plan Examples

1. Conditional
Pressure on
Brake & Throttle

2. Synthesis
Power from
RPMs & Torque

3. Conversion
Device binary or
text data stream
to VSS signal



Why is it important?

- Prevents false interpretation of data
- Provided additional type checking on data transformations
- Ensures robust software defined systems
- Can prevent catastrophic failure when data is used for control

What does it mean?

- Every value is accompanied by its unit of measurement
- This includes boolean values, which should be tristate: on, off, or unknown

Reference

- <https://jcp.org/aboutJava/communityprocess/mrel/jsr385/index2.html>

Next Steps



- Start a subgroup in the Eclipse Sparkplug Working Group for Vehicle Data
- Propose Extension to Sparkplug for VSS
- Define VSS Mapping to Sparkplug Metrics
- Extend Rebirth Message to support sending a Data Acquisition Plan
- Defined the format for this Plan
- Discuss Vehicles (Nodes) could be addressed

Summary

- VSS is a great contribution to interoperability for automotive data collection
- Use industrial standard data transmission protocols: Sparkplug
- Vehicles should be selfaware (maintain their own VSS description)
- A flexible means of selecting, preprocessing, and sending data is needed
- Signal data should always carry its unit of measurement
- MQTT ACL should be used for securing communication
- Do not reinvent the wheel!



Simplify Edge-to-Cloud

aicas. embedded. connected.

Dr. James J. Hunt
Cofounder, CEO, and CTO

aicas GmbH
76131 Karlsruhe, Germany
www.aicas.com
+49 721 66 39 68-0



Leveraging Sparkplug for efficiently transmitting VSS data between the vehicle and the cloud

How sparkplug can be used to transmit VSS data between the vehicle and the cloud.

2:00 PM – 2:25 PM Wednesday



Sparkplug is already widely used in industrial automation as a standard for transmitting data between cloud services and devices. Based on MQTT and ProtoBuf, most of the challenges of flexible data transmission have already been solved. VSS maps naturally onto Sparkplug as a new profile. Combined with data access plans, full control over data collection can be provided with an efficient and robust solution.

Dr. James J. Hunt
CEO & CTO - aicas GmbH

Related Material

Memory Safety

Software Selfawareness: Tracing Software Providence



UN

Global Regulations

ISO

Guidance and Best Practice

Cyber Security and Software Updating

- UNECE r155—vehicle cybersecurity and cybersecurity management systems
- UNECE r156—vehicle software updates and software update management systems

Road Vehicles

- ISO/SAE 21434:2021
Cybersecurity engineering
- ISO 24089:2023
Software update engineering

US Government: EO 14028—SECURING THE SOFTWARE SUPPLY CHAIN

Memory Safety

Why is it important?

- ~70% of Security vulnerabilities are due to lack of memory safety.
- Found in iOS, Android, and Microsoft Products

What does it mean?

- Buffer overrun or out of bounds array reference
- Reference object with wrong type
- Use number as pointer
- Reference object that has been freed

References

- <https://www.memorysafety.org/docs/memorysafety/>
- https://media.defense.gov/2022/Nov/10/2003112742/-1/-1/0/CSI_SOFTWARE_MEMORY_SAFETY.PDF
- <https://www.whitehouse.gov/wp-content/uploads/2024/02/Final-ONCD-Technical-Report.pdf>

Objectives							
Technique	Unambiguous Reference	Fragment Avoidance	Timely Deallocation	Reference Consistency	Deterministic Allocation	Atomic Move	Sufficient Memory
Object Pooling	AC	AC	AC	AC	MMI	N/A	AC
Stack Allocation	AC	MMI	MMI	AC	MMI	N/A	AC
Scope Allocation	MMI	MMI	MMI	AC	MMI	N/A	AC
Manual Heap Allocation	AC	?	AC	AC	MMI	MMI	AC
Garbage Collection	MMI	MMI	MMI	MMI	MMI	MMI	AC

AC = application code, MMI = memory management infrastructure, N/A = not applicable, and ? = difficult to ensure by either AC or MMI.

Related Material

Framework for Dynamic Software Deployment and Management

Framework Requirements

SOA

Flexibly Combine Capabilities

Memory Safety

Exact Garbage Collection

Realtime Scheduling

Properly Prioritize Tasks

Life Cycle Management

Full Control of Software

Software Signature Verification

Only Run Validated Software

Versioned Service Resolution

Prevent Service Mismatching

Low Latency

Minimize Context Switch and Serialization

Adapting Java for OT Systems

Fairness

Conventional Java

- Base Language
- Undefined Scheduling: assumes fair scheduling
- Relies on JiT for performance
- Little support for interacting w/ hardware

Priority

Realtime Java

- Refined semantics & additional APIs (RTSJ)
- Defined Scheduling: preemptive priority & time-sharing (fair)
- Uses AoT for performance
- Support for events, device access, & interrupts

Advantages of OSGi

+ Nanoservices

+ Service-Oriented Architecture

+ Message Based

- > Very small Modules (< Containers)
- > Remote controlled deployment and update
- > Life-cycle management w/ version tracking
- > Service Text
- > Standard Typed Data Bus

Problem

System Lockup

- CPU Exhaustion
- Memory Exhaustion
- RT Scheduling makes this worst
 - Priority Preemption

A high priority thread can block all lower priority threads

Solution

Resource Enforcement

- Resource Enforcement (limits)
 - CPU Use
 - Memory Use
 - Thread Creation
- Safe bundle force termination
 - Thread.stop is unsafe
 - Ensure that finally clauses can run
 - RTSJ: Asynchronous Task Termination
- RTSJ 2.0 provides infrastructure
 - javax.realtime.control
 - javax.realtime.enforce

Framework Comparison

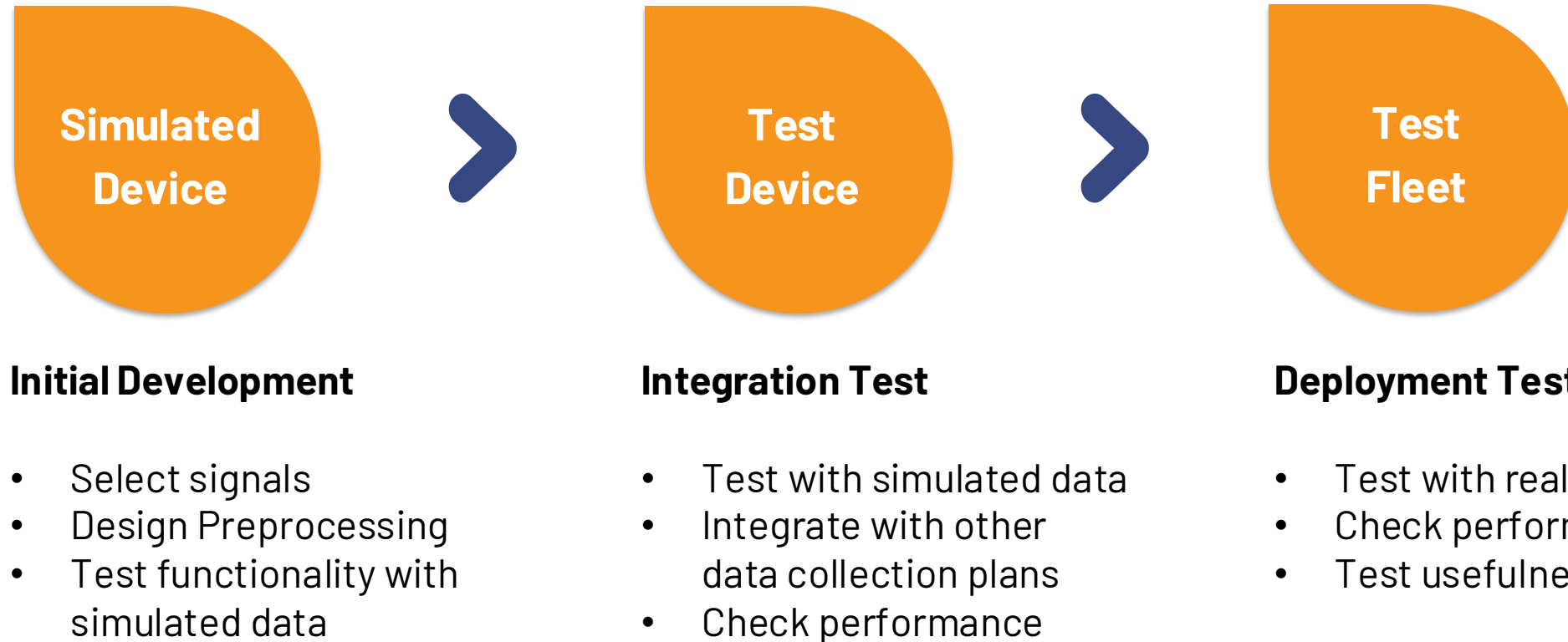


	Memory Safety	Realtime Scheduling	Software Validation	Service Oriented Architecture	Life Cycle Management	Versioned Service Resolution	Low Latency
AUTOSAR Adaptive	Red	Green	Red	Green	Green	Red	Green
Macchina.io	Red	Green	Red	Green	Green	Red	Green
Container	Red	Light Green	Light Green	Yellow	Light Green	Light Green	Red
Conventional OSGi	Green	Red	Green	Green	Green	Green	Green
Realtime OSGi	Green	Green	Green	Green	Green	Green	Green

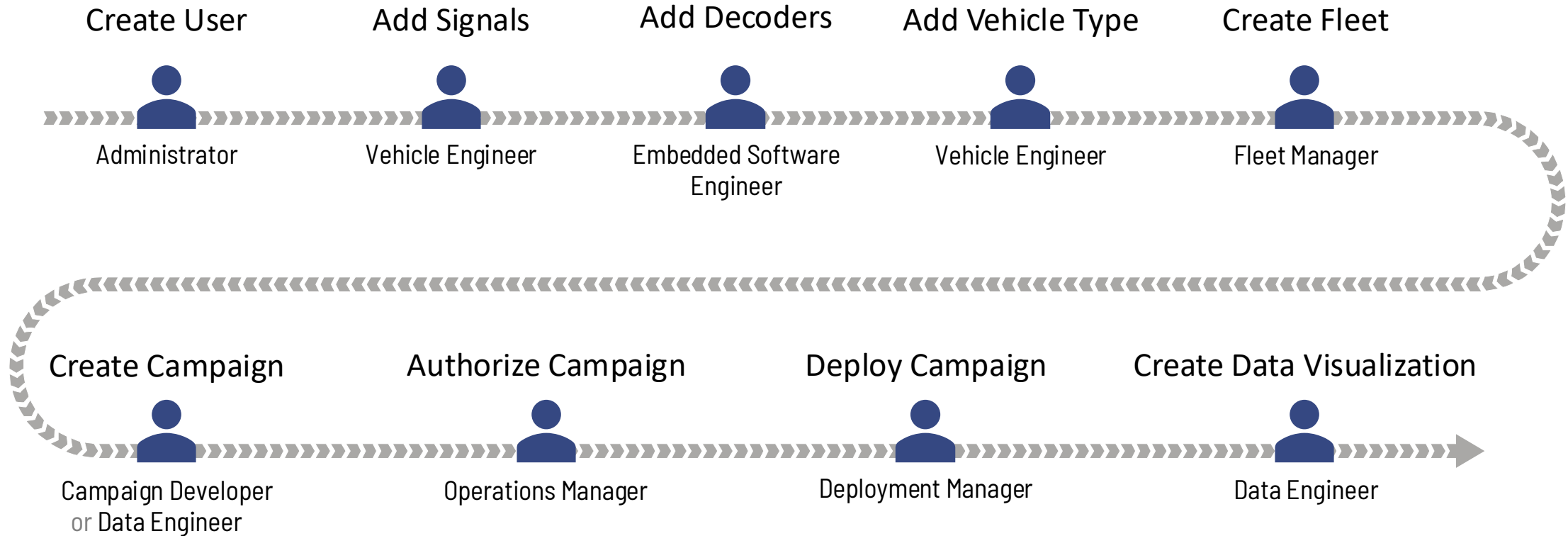
Extra Material

A Business Perspective

Evolution of Data Collection



Role-Based Access UI



Who needs Data Collection? Example Roles and Permissions



Acquisition Element	Create	Read	Use	Authorize	Update	Delete
Vehicle connector	ESW Eng	ESW Eng SW Eng	ESW Eng SW Eng	QA	ESW Eng	ESW Eng Ops Mgr
Enterprise VSS catalog	Vehicle Mgr	Vehicle Mgr SW Eng	Vehicle Mgr SW Eng	QA	Vehicle Mgr	Ops Mgr
DB Schema	IT Eng	IT Eng SW Eng	IT Eng SW Eng	QA	IT Eng	Ops Mgr
Vehicle Data	Vehicle Eng	Vehicle Eng Data User	Vehicle Eng Data User	Vehicle Mgr	Vehicle Eng	Vehicle Mgr Ops Mgr
Vehicle Definition (VSS)	Vehicle Eng	Vehicle Eng Data User	Vehicle Eng Data User	Vehicle Mgr	Vehicle Eng	Vehicle Mgr
Fleet	Fleet Mgr	Fleet Mgr Data User	Fleet Mgr	Deploy Mgr	Fleet Mgr	Fleet Mgr
Data Acquisition Plan	Data User	Data User	Data User	Deploy Mgr	Data User	Data User
Campaign	Data User	Data User	Data User	Ops Mgr	Data User	Data User
Visualization Template	UI Design	UI Design Data User	UI Design Data User	Data Mgr	UI Design	UI Design