Product Development

FNV2 In Vehicle SOA Middleware (FNV2-SOA) (v1.0)

Version 1.0

Version Date: Aug 15, 2017

UNCONTROLLED COPY IF PRINTED

FORD CONFIDENTIAL

Table of Contents

1. Glossary 3
2. Summary 3
3. Alternatives 4
4. Interfaces 4
5. High Level Design 5
5.1. Block diagram 5
5.2. SOA middleware API 6
5.3. Topic and message structure 7
5.4. Messaging pattern 7
5.4.1. Blind Broadcast(one to many) 8
5.4.2. On Broadcast 8
5.4.3. Request/Response (RPC) 8
5.4.4. System messages 8
5.5. Service Manager 8
5.6. Security 9
5.7. Message priority 10
5.8. Broker crashes and restarts 10
5.8.1. SOA 11
5.9. ECG replacement 13
6. Performance 13
7. Dependencies and Risks 13
8. Questions 13
9. References 14

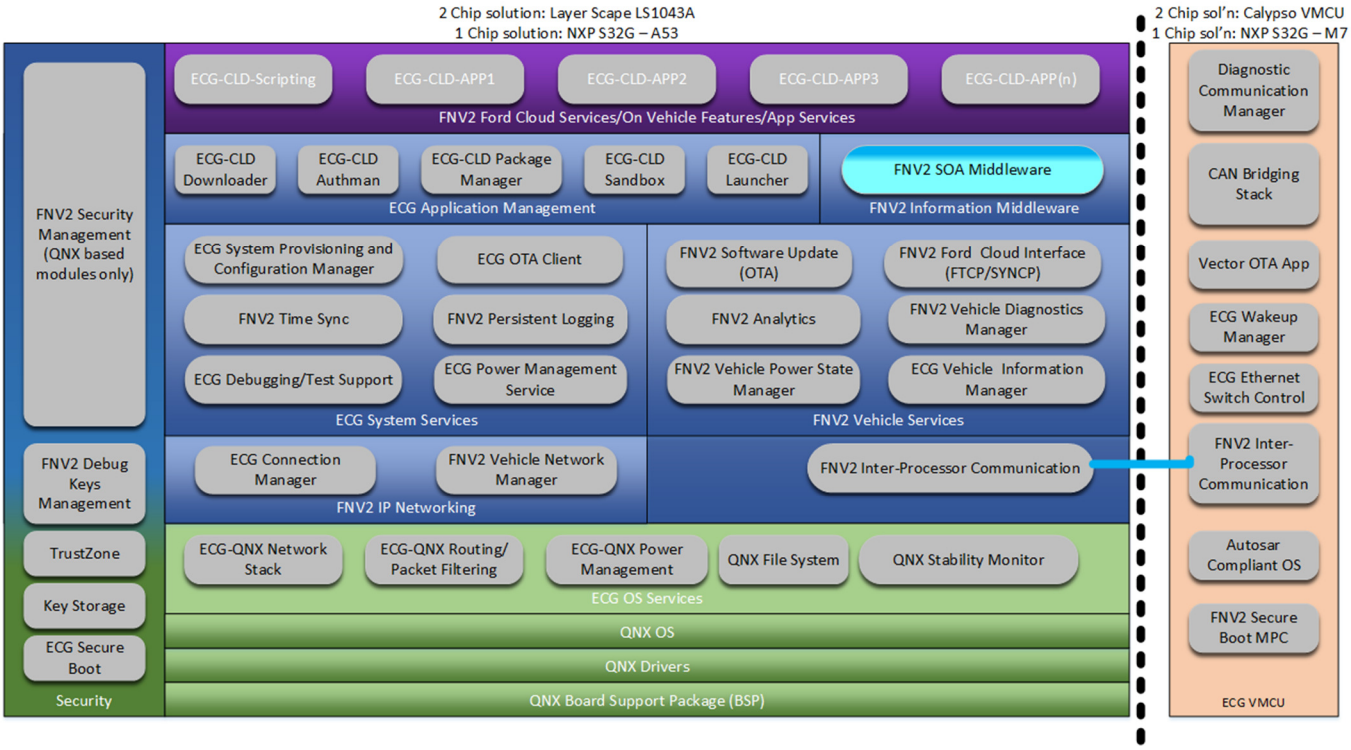
1. Glossary

	Description
ACL	Access Control List
SOA	Service Oriented Architecture
DDS	Data Distribution Service
MQTT	Message Queuing Telemetry Transport
TLS	Transport Layer Security
ECG	Enhanced Central Gateway
FTCP	Ford Telematics Control Protocol
vSOME/IP	Scalable service-Oriented MiddlewarE over IP
RPC	Remote Procedure Call
ECG-FCI	Ford Cloud Interface
ECG-VIM	Vehicle Information Manager

2. Summary

The SOA middleware is the layer in charge of providing a framework for local and remote components to exchange control and data messages in a secure and efficient manner. The typical services provided are:

- Service Registration: This block allows to register FTCP command handlers and to keep the status of downloadable services.
- Encryption: For remote clients located outside of the ECG, TLS encryption is mandatory. This library includes the main cryptographic algorithms and other facilities (e.g. OpenSSL).
- Authentication & authorization: For local clients, there is no encryption required and a simple authentication mechanism is sufficient based on uid/gid. Also, once a client is authorized to connect, the SOA checks if the client is authorized to get access to the requested topic (ACL).
- Queue: It may be required to queue some of the messages event if the client is not connected so once it connects, the relevant messages can be delivered.
- Routing: This block includes the mechanism required to direct messages between the client and the service, including filters
- Transaction logger & Telemetry: Various events need to be reported as connected clients, permission denial, etc



3. Alternatives

Three FNV2 SOA frameworks have been investigated:

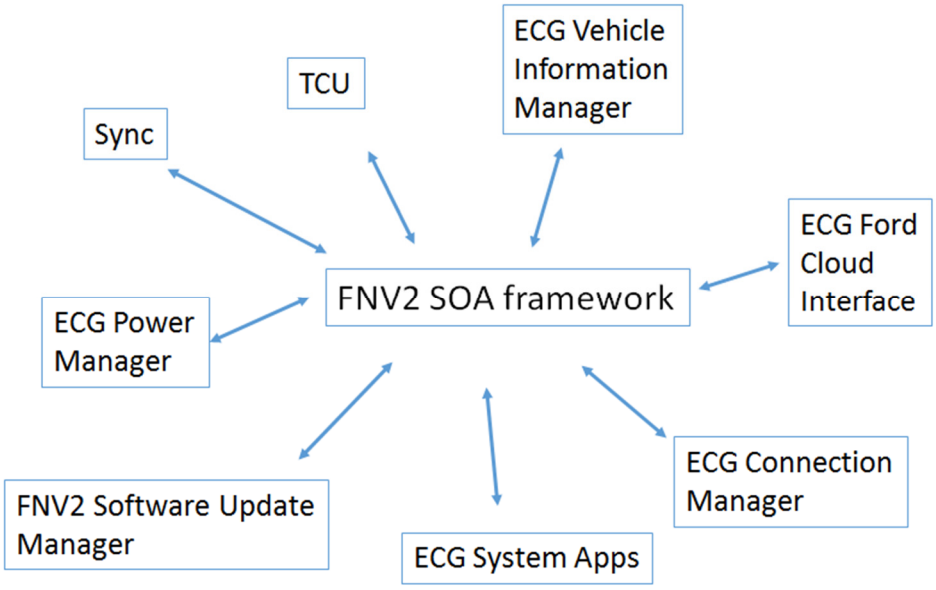
vSOMEIP which is the standalone version supported by Genivi found in AutoSar.

MQTT already used to manage FTCP command in the cloud.

DDS, the brokerless and distributed system

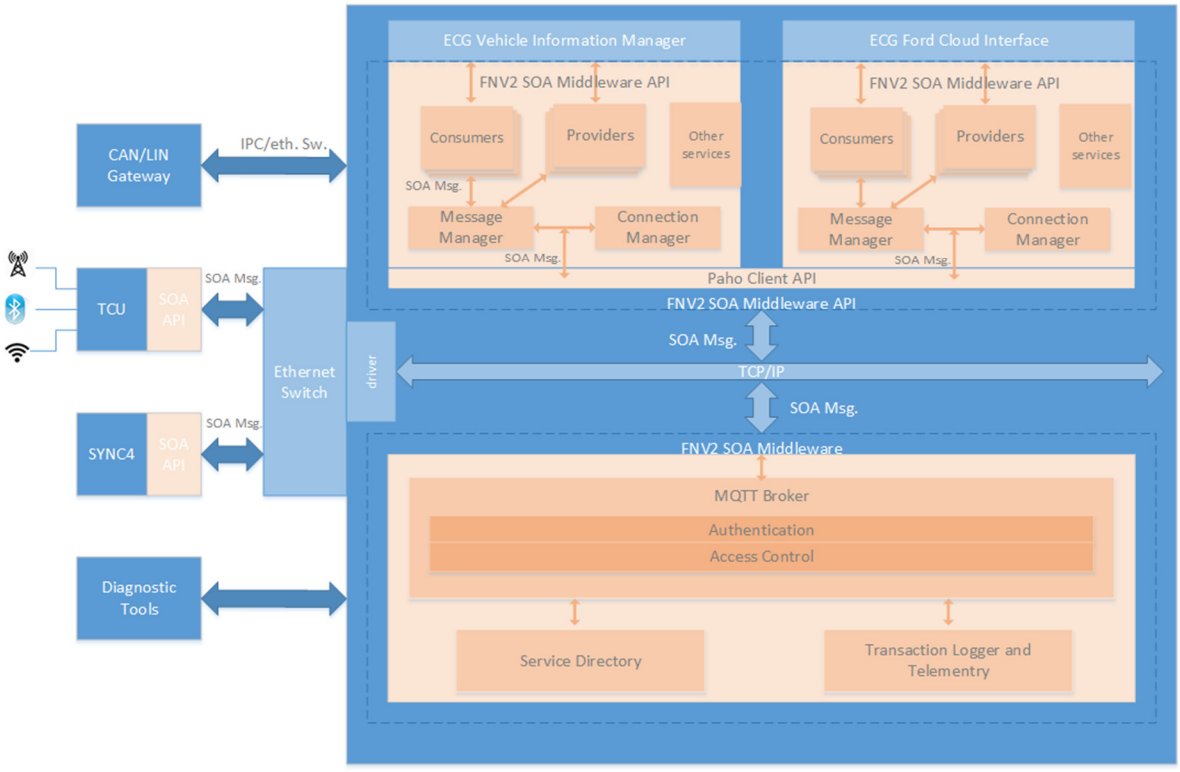
4. Interfaces

The FNV2 SOA middleware is the main communication backbone. The ECG software components and some of the hardware modules outside of the ECG rely on this framework to exchange messages using the publish subscribe messaging pattern. A subset of the clients of the framework are represented below.



5. High Level Design

5.1. Block diagram

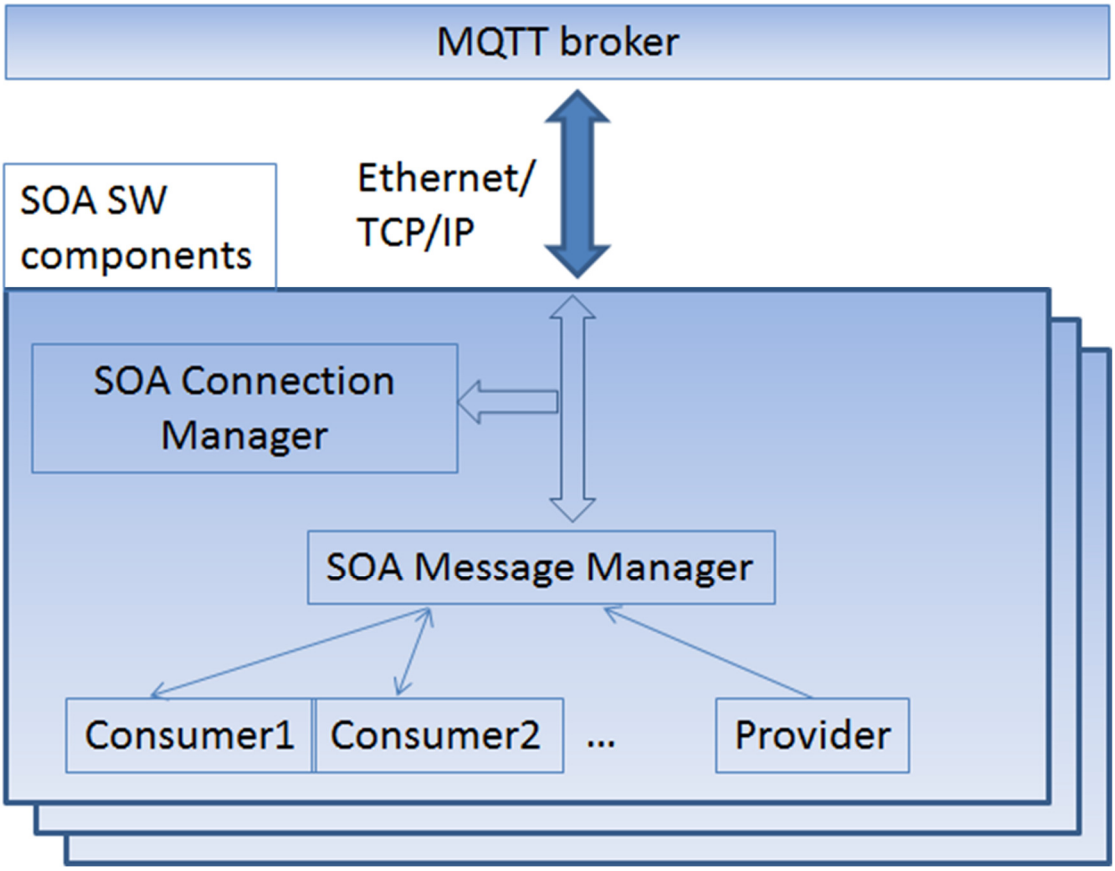


5.2. SOA middleware API

The MQTT protocol is implemented through the Paho library. However, this API does not provide some of the functionality the SOA framework requires as Remote Procedure Call mechanism. A wrapper is then implemented on top of the Paho API which has the following benefits:

- Abstract the specifics of the MQTT Paho library. If anything will change below this layer (MQTT broker, different client interface, etc.) the SOA middleware clients will not have to be changed as they will see the same interface
- Extend the Paho library with additional features as RPC
- Shield the SOA middleware clients from the complexity of the topic structure. This topic structure is used to communicate over MQTT to publish and subscribe. Given the various requirements and the large number of topics, the API will manage this task rather than leaving each clients doing their own specific management.
- Provide a more robust and reliable interface implementing additional verification and validation of the settings. The Paho API can be used to change a lot of settings of the publish-subscribe mechanism. Some of them may be not allowed and will be enforced by this additional wrapper.
- Support other transport mechanisms as IPC to support security features as TLS sessions with Sync or TCU and client authentication.

A MQTT client is an ECG software component interacting with another component through the MQTT broker. This MQTT client is called SOA component. It can be either a service consumer a service provider or both. The figure below is a representation of a SOA software component which includes multiple clients and one server.



The SOA connection Manager is the direct interface with the MQTT broker. This layer is in charge of providing all the infrastructure to manage the connection with the broker. Connection settings and connection callbacks (connection failed, disconnection) are located in this layer.

The SOA Message Manager is in charge of receiving the incoming messages from the other SOA SW components and routing those messages to the appropriate client . Also, when a client sends a message to the broker, this is done through this interface.

The communication between the clients and SOA Message Manager can be uni-directional if the client is just listening for a particular message or can be bi-directional if the client is requesting a response to a request (RPC).

5.3. Topic and message structure

See: FNV2-SOA: MQTT Topic and Message Structure Document

5.4. Messaging pattern

There are two types of communication mechanisms to be supported by the SOA middleware: broadcast and RPC

Additional details in FNV2-SOA see: MQTT Topic and Message Structure Wiki/Document

5.4.1. Blind Broadcast(one to many)

Broadcast is defined as a client or service publishing a piece of information without knowing if anybody subscribed to it. There is no answer expected in that case. This is a typical use of the publish-subscribe messaging pattern.

The minimum topic structure for this purpose is:

"SERVICES/DATA/..."

Any clients interested in knowing information from a particular service provider will subscribe to this topic.

5.4.2. On Broadcast

Blind Broadcast as described above can be a waste of bandwidth if no client are listening to the broadcast topic. The on-demand broadcast requires the client to explicitly request the provider to broadcast the data. If the client disappears gracefully or ungracefully, the broadcast will stop.

5.4.3. Request/Response (RPC)

When a client wishes to send a request to a destination and expects a response (1 to 1), the following topic structure should be used:

"SERVICES/REQUEST/PROVIDER_ENDPOINT"

where:

PROVIDER_ENDPOINT is the destination of the message e.g. service providers on SYNC, TCU, etc.

The request message contains the response topic to let the provider knows the response destination.

Response topic example:

"SERVICES/RESPONSE/CONSUMER_ENDPOINT"

5.4.4. System messages

Those messages are targeted to the broker and not intended to be received by any subscribers.

5.5. Service Manager

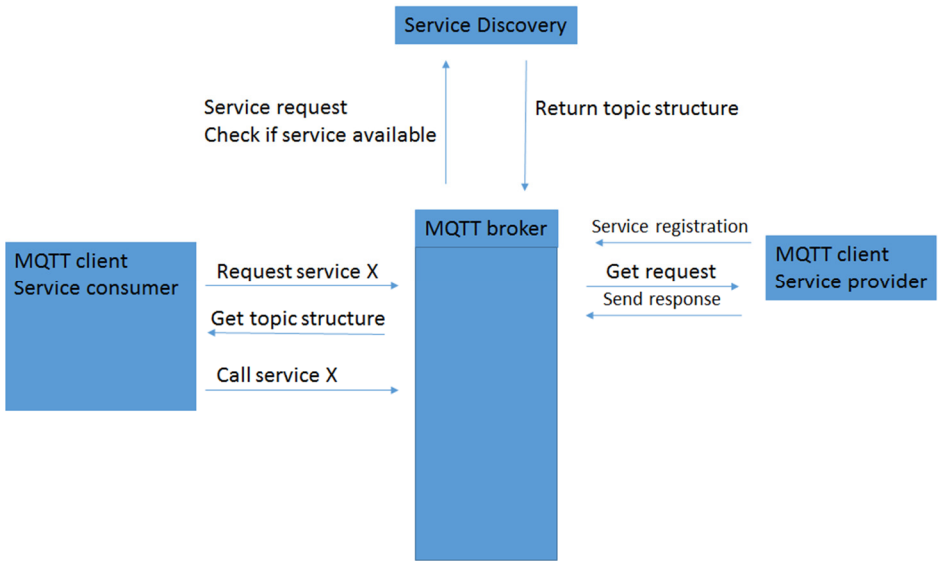
Service Manager maintains a mapping between FTCP command names and handler endpoints. This is used by the FCI module for routing FTCP commands to the appropriate handler.

Service Manager contains also a database of the topics of all the FNV2 services available and the corresponding permission for each consumer. This allows to restrict access to services to authorized clients as part of the ACL mechanism described in the SOA Security Architecture Document/Wiki.

Note the below description is a proposal for future functionality to support service discovery :

Any additional details as the service version or state could be hidden in the structure of the payload to prevent to have too many topics. However, there is a trade-off between topic complexity and number of messages received by a subscriber. If the subscriber is only interested in a specific service, it will have to subscribe to all the messages broadcasted by a components and examine the payload to figure out if this is the one of interest.

Given the complexity and the use case variants, the topic structure management is implemented in the SOA middleware API. The intent is to expose a list of services to the clients and hide the underlying topic structure to get access to the service. This is typically the task of the Service Discovery to map service and topic name in a central place using a database.



5.6. Security

For external components connected to the SOA as SYNC and TCU, a TLS session is setup. For ECG clients, only authentication/authorization will be sufficient.

MQTT implementation already contains some security related features as user name and password authentication at connection time, topic-based ACL and TLS sessions. However, those mechanisms do not provide any options to uniquely identify clients. Some ECG components as the connection manager has to make sure a client requesting to setup a WiFi, Cellular or Bluetooth connection has the appropriate permissions to do so. The details of the current proposals are described on the following pages:

Security Architecture

5.7. Message priority

For analytics or debug purpose, we may have to send to the Cloud CAN Com signals which could represent a significant amount of bandwidth. To make sure there is no impact on the other messages flowing through the SOA framework, message priority has been proposed.

This feature has not be planned and developed at this point.

5.8. Broker crashes and restarts

The main component part of the SOA framework is the MQTT broker. If the broker crashes, mainly 2 options are available:

- **Restart the broker**

The broker can be restarted and can restore some of it state preceding the crash. This is done through persistence capability. The SOA Client API has also a feature to detect connection lost and reconnect automatically. Those two features are the minimum requirements to be able to restart the broker after a crash.

-
- Persistent session

When a SOA client connects to the broker, a persistent session can be set. It means that the client subscriptions are stored by the broker in a database and reloaded when the broker is restarted. So the client does not need to subscribe again and will continue to receive the publications the same way as before the restart. The persistent session is usually sets by the consumers only as there is no advantage for a service provider to do it. When a client connects, the broker returns whether a previous persistent session was saved. If so, the client does not need to subscribe again.

The interval between each time the in-memory database is saved is configurable. It is also possible to trigger the backup after a given number of subscription change. Using this mechanism, it is then possible to save every subscription and not loose them when the broker restarts. However, for performance reason, the overhead may be to high to support this scenario if the subscription rate is too high. Also, as this point, there is no details about unsubscribing. We would expect the broker to remove the subscription from the database and the client does not keep receiving the publishes once the broker restart. An lastly, this is potentially a memory leak.

-
  - Connection lost detection

If the broker crashes, the SOA client API is going to be notified wherever the client is located (ECG or Gateway running on fast ECUs). The notification delay is what is defined as the KeepAlive interval. If no message is exchanged with the broker within this interval, the SOA API will send a ping request to the broker. If no response is received, the connection lost event is asserted. If the connection is lost or

broker is restarted in the middle of a RPC transaction, the client which initiated the request will get a timeout and will have to retry. The SOA framework will keep trying reconnecting without the intervention of the client.

- **ECG restart**

That scenario is simpler to manage. As all the ECG will be reset and any existing broker database file corresponding to previous persistent sessions should be deleted.

For the external modules as Sync and TCU, the SOA Gateway will keep trying to reconnect to the ECG/Broker as no specific signal may be available to notify that the ECG is down. The number of retries and for how long the retries will happen will need to be determined based on the maximum restart time of the ECG.

For both scenarios, once the broker is initialized, the ACL database is reloaded and the broker starts listening to port 1883 (default value).

Restarting the broker without the need to restart the whole ECG is the preferred option. As this will require more features to be developed, it is recommend to implement the ECG restart solution first and to migrate to the broker restart once performance analysis is done and all the required features are made available.

Irrecoverable software faults will be reported by Stability Monitor. SM will take corrective action specific to the fault. This could result in a reboot of the ECG or ultimately a DTC being raised.

5.8.1. SOA

DTC (Hex)	DTC (display)	DTC Type	Root Description	Failure Type Byte Description	Used by components	Required action	Comments	Proposal for Implementation
			Connection failure/ broker crash	**Recovery:** Stability Monitor restarts broker **Corrective**: If recovery fails, re-attempt to restart broker **Failure:** Issue DTC to notify that SOA broker is not functional and cannot be recovered.	Any SOA component		SOA broker crashes and will prevent any soa components to communicate.	I would assume that this either requires a module reflash, or is due to a HW issue to correct. If yes, then I would set against appropriate U3000 sub-type.

			ACL database corrupted	**Recovery:** none **Corrective**: none **Failure:** Issue DTC to notify that ACL database is corrupted	SOA Service Manager and SOA broker ACL plugin		The ACL database is needed to define for each services, the access right. If this database is corrupted, none of the service will be accessible	I would assume that this either requires a module reflash, or is due to a HW issue to correct. If yes, then I would set against appropriate U3000 sub-type.
			Service registration file corrupted	**Recovery:** none **Corrective**: none **Failure:** Issue DTC to notify that service registration is corrupted	SOA Service Manager		This xml file is used to register service to the Service Manager. If the file is missing or corrupted, the service cannot be registered	I would assume that this either requires a module reflash, or is due to a HW issue to correct. If yes, then I would set against appropriate U3000 sub-type.
			Service Manager crash	**Recovery:** stability monitor to restart process **Corrective**: if recovery failed, retry to start the process **Failure:** Issue DTC to notify that service manager is non functional	SOA Service Manager		If the service manager is not functional, service cannot register and FCI will not be able to map commands to service endpoints.	I would assume that this either requires a module reflash, or is due to a HW issue to correct. If yes, then I would set against appropriate U3000 sub-type.
			Communication failure with ALM	**Recovery:** restart IPC interface **Corrective**: If the recovery failed, try to restart Service Manager **Failure:** Issue DTC to notify that service manager communication is broken	SOA Service Manager		The IPC interface is needed for ALM to send information on app install	I would assume that this either requires a module reflash, or is due to a HW issue to correct. If yes, then I would set against appropriate U3000 sub-type.

			Component failed to connect to broker	**Recovery:** component will try to automatically reconnect if the first connection fails. **Corrective**: none **Failure:** Issue DTC to notify that component was not able to communicate with broker if all reconnection failed.	SOA components		SOA component is not able to connect to the broker. As the component may only have the SOA interface, no communication is possible with the rest of the system	See notes for invalid device in Networking Section.

5.9. ECG replacement

ECG is powered down, disconnected and replaced by a new module which is powered up. The broker should not have any database files and only the broker configuration file is required. This file is part of the system image. This scenario is the same as the ECG restart described above.

6. Performance

For the current implementation. most of the clients connected to the SOA are low bandwidth as the communication is mainly for control and signaling purposes. The highest contributor identified is the Vehicle Information Manager which may publish either large number of messages or messages with a significant payload size (few kB). This high activity could happen mainly for analytics/debug purposes when for instance all the 100ms CAN signals are requested.

MQTT benchmark results are located: MQTT benchmarks

7. Dependencies and Risks

8. Questions

No.	Description	Owner	Status
	Question description here	*Current owner, if assigned*	Current status

9. References

Interprocess Communication (IPC):

- D-Bus: https://en.wikipedia.org/wiki/D-Bus
- QNX message passing interface: http://www.qnx.com/developers/docs/6.4.1/neutrino/sys_arch/ipc.html
- D-BUS implementation in several languages: http://dbus.freedesktop.org
- MQTT machine-to-machine (M2M)/IoT connectivity protocol: http://mqtt.org/
- Android IPC/Binder Framework: https://events.linuxfoundation.org/images/stories/slides/abs2013_gargentas.pdf
- Autosar SOME-IP performance: https://www.realtimeatwork.com/wp-content/uploads/SAE-SomeIP-web.pdf
- Data Distribution Service: Presentation