FNV2 Connectivity Manager Architecture (FNV2-CM)

Status	IN PROGRESS
Completion (%)	10%
Version	0.3
Owner	Brar, Karmindar (K.) Marupaka, Nagendra (N.) Park, Seung (S.D.)
Reviewer	Lisak, Darko (D.) Murugesapillai, Nava (N.) Ma, David (D.) Alievsky, Michael (M.) Puri, Ron (R.)
Review Date	v0.1 - 22 Mar 2017 v0.2 - 23 Mar 2017
Approver	Murugesapillai, Nava (N.)
Approve Date	v0.1 - 22 Mar 2017 v0.2 - 31 Mar 2017
HLD Jira	ECG-10031 - Authenticate to see issue details

Requirement JIRA Links	
Test Story Link(s)	ECG-10031 - Authenticate to see issue details
Test Plan Link	Test Plan: Connection Manager

1. Glossary

Term	Definition
ECG	Enhanced Central Gateway
ECU	Electronic Control Unit
CAN Bus	Controller Area Network Bus
FNV2-CM	FNV2 Connectivity Manager
FNV2	Fully Networked Vehicle 2nd Generation
FNV2-FCI	FNV2 Ford Cloud Interface Module
FTCP	Ford Telematics Communication Protocol
IVDCM	In-vehicle distributed connectivity manager
GPB	Google proto buf
QWF	QNX Wireless Framework
SOA	Service Oriented Architecture
SDL	Smart Device Link
SDN	Service Delivery Network
TCU	Telematics Control Unit: the cellular modem
OTA	Over-The-Air Software
WIR	Wireless Interface Router

An extensive list of the terminology used throughout the Silver project documentation can be found here: Terminology

2. Summary

2.1. Background

The previous generations of Ford's Fully Networked Vehicle employed a Smart Data Link Connector (SDLC) as the gateway/hub for interconnection between various ECU's. CAN bus was the primary transport mechanism used, with IP connectivity limited to SYNC and TCU only. As such, the requirement for Connectivity Management was distributed across these two ECU's.

The Enhanced Central Gateway (ECG), the successor to SDLC embodies the advanced networking model that has been introduced in Ford's Fully Networked Vehicle 2 (FNV2) project. ECG controls and enables enhanced Ethernet capable modules interconnected to provide faster and cheaper data transfer mechanisms. A centralized FNV2 Connectivity Manager (FNV2-CM) is instrumental in achieving the high data rates and for providing seamless Ethernet connectivity to all the modules, applications and services that reside on the vehicle.

The architecture presented in this document proposes an FNV2-CM master module residing on the ECG with "thin" FNV2-CM entities residing on other ECUs connected to Ethernet, collectively referred to as Wireless Interface Router (WIR). This approach ensures that the networking state machines on the ECG and other Ethernet-connected ECUs are in sync. Moreover, this enables applications residing on any Ethernet-connected ECUs to have access to available edge network interfaces for establishing data connections to cloud/internet.

2.2. Purpose

This document describes the overall architecture and the design motivations for the Connectivity Manager module in the Networking component of the Enhanced Central Gateway (ECG) : ECG-440.

The following sections cover the requirements that were instrumental in drafting the design, the typical use cases, the state machines employed and the interfaces and interactions between FNV2-CM and various FNV2 modules. Finally, future considerations and risks are provided.

2.3. Scope

The FNV2 Connectivity Manager (FNV2-CM) controls and manages the IP data transport interfaces (eg. WiFi, Cellular) over which connections are made between the vehicle and external networks. In essence, FNV2-CM:

1. maintains a single list of available IP data transport interfaces and their priority and related QoS, bandwidth, latency and cost
2. provides API for getting the IP data transport interfaces list
3. notifies apps about IP data transport interface status changes
4. maintains traffic statistics on all the IP data transport interfaces
5. will provide means to choose the lowest cost route and avoid single point of failure i.e. have a multiple routes
6. will make the route selection will based on application intent/policy provided by an application or service (FNV2-FCI, OTA etc)

Note that applications are responsible for establishing the end-to-end connections over the IP data transport interfaces. A vehicle service or application wishing to send/receive data over a connection to an element external to the vehicle needs to request an IP data transport interface from the FNV2-CM. Note that the service or application may reside on a module other than the ECG. Authorization and authentication of Network Interface Allocation Requests is performed by a combination of policy enforcement and ACL mechanism.

3. Requirements

Current requirements identified for FNV2-CM are captured on the FNV2 Connectivity Manager Requirements Analysis page. These requirements are listed below and on the ECG Connectivity Manager (CM) and Networking dashboard.

Key	Summary
Authenticate to retrieve your issues	

No issues found

4. Interfaces

As a critical component in the networking architecture in the ECG, FNV2-CM interacts and interfaces with various modules and software components in ECG and in other ECU's (TCU and SYNC). FNV2-CM can use the following two mechanisms to manage these interactions:

1. Use FNV2-SOA middleware to publish/subscribe to various topics across all the ECU's and components.
2. Provide local FNV2-CM entities and API's that will enable communication with other services and components.

Two models of interface architecture are proposed based on the above two mechanisms.

4.1. Model A: FNV2-SOA Centric Model

In this architecture, FNV2-SOA middleware is employed extensively by the FNV2-CM to communicate with the data connection providers on the SYNC (USB, Bluetooth, AppLink and WLAN) and on the TCU (Cellular and WLAN). The FNV2-CM clients residing on the two ECU's are used primarily for handling requests from local applications and services and to interface with the local networking stack (QNX on SYNC and Linux on TCU). The FNV2-CM (as opposed to FNV2-CM clients) is responsible for all the connection management functionality. The connection interactions (bring-up and tear-down) happen directly between FNV2-CM and local interface connection providers over FNV2-SOA.

Applications and services internal to the ECG can communicate with the FNV2-CM either over FNV2-SOA or local IPC mechanism provided by host the OS (QNX), contingent on the performance comparison between MQTT and local IPC. MQTT is the preferred mechanism as it removes the dependency on the host OS, thus allowing for application portability to a different ECU (SYNC, TCU etc.) in the future.

Figure 1 provides an overview of the architecture.

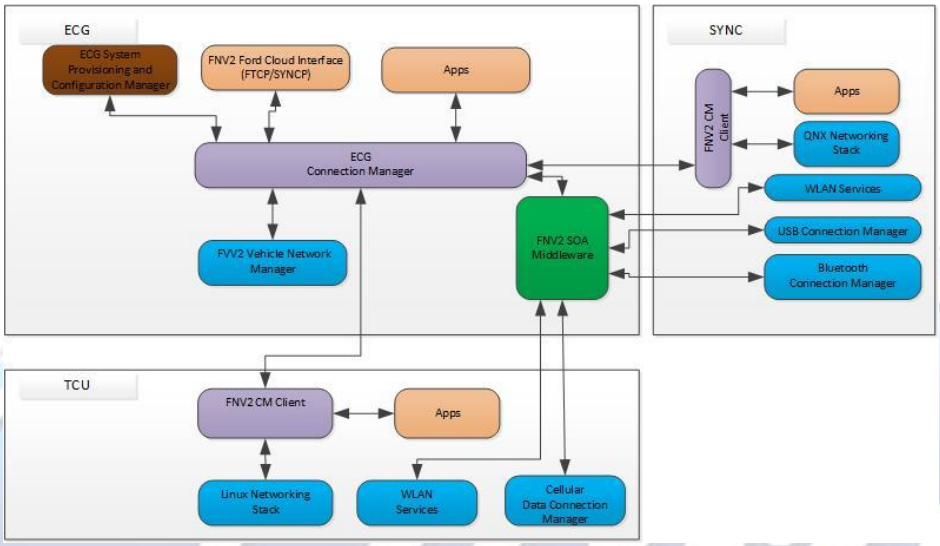

Figure 1

4.2. Model B: FNV2-CM Client Centric Model

In this model, the FNV2-CM clients residing on the SYNC and TCU are the only interface mechanism available for the local applications, services and components if they need interact with the FNV2-CM module. All the connection requests, networking stack interactions and connection bring-up, tear-downs etc., are routed through the FNV2-CM clients. FNV2-SOA is not involved in the communication between services, applications, FNV2-CM, FNV2-CM etc.

This model mandates that the FNV2-CM clients maintain a connection state machine and that it be in sync with the state machine of the FNV2-CM.

Figure 2 provides an overview of this architecture.

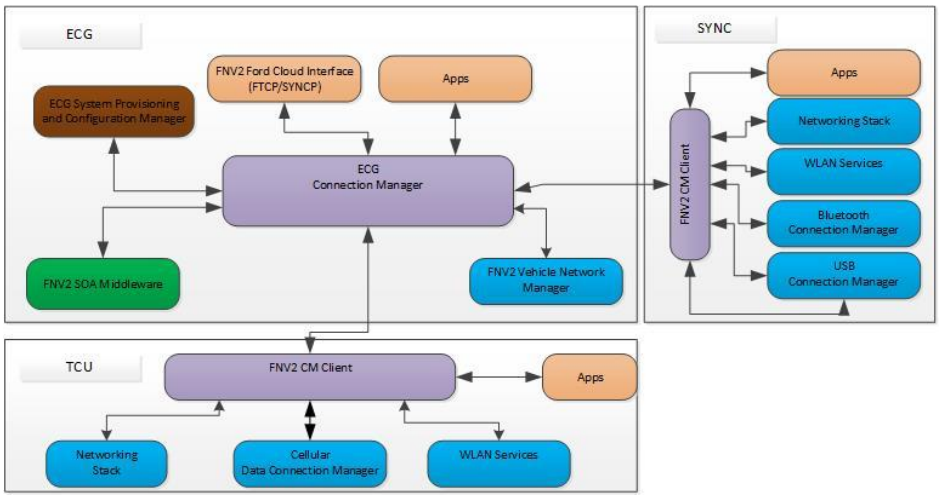

Figure 2

Model A shown in **Figure 1** is the preferred model for the FNV2-CM interface architecture due to the following advantages:

1. Since the connection states are maintained in one place (FNV2-CM), there is no chance of connection state machines being out of sync. Additionally, this reduces the synchronization overhead between FNV2-CM and FNV2-CM clients.
2. A centralized communication system between FNV2-CM, its clients and other modules over FNV2-SOA is in conformity with the current security stipulations proposed for FNV2.
3. In this model, the applications residing on the SYNC and TCU employ the respective FNV2-CM clients as surrogate brokers into the MQTT framework of the FNV2-SOA.

4.3. Module Interfaces

Below is a list of interfaces that FNV2-CM will need to maintain with various components:

1. **ECG-NET-Ccm** (*ECG System Provisioning and Configuration Manager* to *FNV2 Connectivity Manager*): This interface allows the configuration of the *FNV2 Connectivity Manager*, e.g. specific criteria for transport selection.
2. ***FNV2-NET-SOAcm*** *(FNV2-SOA Middleware to FNV2 Connectivity Manager)**:*** When higher-layer components require a connection to an external network, they will make the request through the FNV2-SOA Middleware. This interface will then allow the FNV2-SOA Middleware to request the connection required by other components. At the same time, the *FNV2 Connectivity Manager* sends a connection request to the TCU via the *FNV2-SOA Middleware* as well. The *FNV2-SOA Middleware* would pass on this request message to the TCU to set up the actual external network connection.

3. **ECG-NET-FCcm** (*ECG Ford Cloud Interface* to *FNV2 Connectivity Manager*): For those connections that require FTCP messages to go to the Ford Cloud, this interface will be used to allow the *ECG Ford Cloud Interface* component to request the connection directly without going through the *FNV2-SOA Middleware*.
4. **ECG-NET-CMnm** (*FNV2 Connectivity Manager* to *FNV2 Vehicle Network Manager*): This interface is required for any lower layer networking requests made by the *FNV2 Connectivity Manager*.
5. **ECG-NET-CMsm** (*FNV2 Connectivity Manager* to *FNV2 Security Manager*): When a connection is requested, security measures need to be taken to ensure that the request is authenticated. This interface provides the *FNV2 Connectivity Manager* with a way to provide the authentication.
6. **ECG-NET-CONN** (Any Ethernet node to *FNV2 Connectivity Manager*): Using the *FNV2-SOA Middleware*, any given (permitted) application on an Ethernet node will be capable of requesting a connection from the *FNV2 Connectivity Manager*. The interface protocol will be administered via the *FNV2 Connectivity Manager Client* library that is present on Ethernet nodes that require connections.
7. **ECG-TCU-CONN** (*FNV2 Connectivity Manager* to TCU): When a connection is requested from the *FNV2 Connectivity Manager*, it will be required to set up an actual connection from the TCU. Again, using the *FNV2-SOA Middleware*, an interface protocol will be in place between the *FNV2 Connectivity Manager* and the TCU to perform this action.
8. **ECG-SYNC-CONN** (*FNV2 Connectivity Manager* to SYNC): When a connection is requested from the *FNV2 Connectivity Manager*, it will be required to set up an actual connection from the SYNC. Again, using the *FNV2 SOA Middleware*, an interface protocol will be in place between the *FNV2 Connectivity Manager* and the SYNC to perform this action.

The next few sections will provide a brief overview on the main interfaces.

4.3.1. FNV2 Ford Cloud Interface (FNV2-FCI)

Based on the application intent the FNV2-FCI will request appropriate connection (cellular, WiFi) from FNV2-CM. CM will provide the list of available connections to the FNV2-FCI. FNV2-FCI can request the bring up of a cellular connection if its not already up.

The Ford Cloud Interface (FNV2-FCI) is responsible for handling FTCP connections to the Ford SDN (Service Delivery Network). This includes

- FTCP messaging between the Ford SDN and cloud
- Requesting a connection to Ford SDN from CM.

The FNV2-FCI is responsible for prioritizing and queuing of commands to Ford SDN, based on the nature of the FTCP request and application intent specified by the the apps. The intents specify the priority, time sensitivity and possible selection of an IP data network interface (TCU Cellular, SYNC WiFi etc.).

FNV2-CM does not process or handle incoming SMS messages.

4.3.2. TCU

4.3.2.1. Cellular Data Connection Manager

FNV2-CM will manage the bring up and tear down of cellular data connections on behalf of clients. FNV2-CM will provide a list of cellular connections available and their current states to the applications.

1. On demand cellular connection bring up and tear down
2. Handling IP Address changes on the cellular network
3. Collecting traffic stats on cellular network connection
4. Turning off and on data connection during eCall
    1. FNV2-CM listens to ECall events and is responsible for tearing down all data connections when an Ecall is started and notifying the application that data is disconnected. FNV2-CM will block all data connection requests during an ECall i.e. no data connection request will be sent to TCU cellular until ECall is over. Once the ECall is over FNV2-CM is responsible for and enabling of data path and sending notification to the application that data path is up.

4.3.2.2. WLAN Services

FNV2-CM will provide the TCU WLAN connection and its current state to the applications.

WLAN service will be provisioned either pre-loaded profiles or through Sync HMI, FNV2-CM plays no role in configuration of WLAN profiles.

1. Requesting STA mode or Hotspot Mode during factory provisioning
2. Collecting traffic stats on WLAN network connection
3. Handling of IP Address changes on the WLAN interface

4.3.2.3. Linux Networking Stack

FNV2-CM will use the Linux Networking Stack to setup GRE tunnels for routing packets through the ECG. FNV2-CM also uses the networking stack to setup packet filters that will only allow authorized applications to access the Ford SDN.

4.3.3. SYNC

4.3.3.1. WLAN Services

FNV2-CM will provide the Sync WLAN connection and its current state to the applications.

WLAN service will be provisioned either preloaded profiles or through Sync HMI, FNV2-CM plays no role in configuration of WLAN profiles

1. Collecting traffic stats on WLAN network connection
2. Handling of IP Address changes on the WLAN interface

4.3.3.2. SYNC Smart Device Link (SDL)

SDL plugin is used to communicate over Bluetooth or USB to a users mobile phone. QNX io-pkt supports creation of TUN interface to create a virtual IP interface for passing IP data to SDL plugin. SDL plugin receives IP packets from the TUN interface created on QNX and forwards the packets to the users mobile phone.

FNV2-CM will manage Bluetooth, USB and WiFi Direct connection connected to mobile phone using TUN interface. FNV2-CM will provide the Sync Bluetooth/USB connection and its current state to the applications.

FNV2-CM does not configure the Bluetooth/USB/WiFi present on Sync module, Bluetooth/USB/WiFi are configured through the Sync HMI,

- Collecting traffic stats on the TUN interface.

4.3.3.3. QNX Networking Stack

FNV2-CM will use the QNX Networking Stack to setup GRE tunnels for routing packets through the ECG. FNV2-CM also uses the networking stack to setup packet filters that will only allow authorized applications to access the Ford SDN.

4.3.4. FNV2-SOA

FNV2-CM employs SOA interface as the mode of communication between ECG-CM and the local FNV2-CM entities residing on other ECUs. More information about the SOA client used by FNV2-CM is provided here. The IDL implementation details are provided here.

4.3.5. Offpeak Cellular Data Usage Management

As part of an effort to streamline data/bandwidth usage costs and data ates

4.3.6. Power Management

4.3.6.1. TCU Power Management

FNV2-CM is not responsible for power management on the modem (TCU). The power management module on the TCU will listen to power states from the FNV2 Vehicle Power State Manager and act accordingly. The modem will not go into a sleep state until all the cellular data connection requests from the FNV2-CM are torn down.The applications are responsible for listening to vehicle power state and informing FNV2-CM to disconnect the connection it

requested, when the vehicle enters power save mode. The modem will only go into power save (sleep) mode when all the data connections have been torn down.

4.3.6.2. ECG Power Management

FNV2-CM will maintain a reference count of all the Apps that are requesting a connection on a particular interface. Once all the connection on a particular interface are torn down it will clean up the connection state machine for that interface. The Apps on the ECG are responsible for listening to the FNV2 Vehicle Power State Manager and sending disconnect request to FNV2-CM.

5. Use Cases and Sequence Diagrams

5.1. Use Cases

The following are the currently documented list of typical use cases involving the FNV2 Connectivity Manager. They provide a view of the interactions and flow between various modules and FNV2-CM. As the design and architecture evolves additional use cases will be added to the document.

5.1.1. Boot Up of ECG

Ignition On - FNV2-CM Bring up and registration.

Actors	FNV2-CM, FNV2-FCI, TCU DCM, Ford SDN, ECG-VNM, FNV2-SOA
Pre-conditions	1. ECG is in deep sleep 2. TCU is in DRX or Wake-up mode
Scenario Description	Driver turns ignition key on and expects the car to be connected to the cloud. For this to be achieved, the ECG boots up and initializes FNV2-CM and its clients. Subsequent to the boot up sequence, FNV2-CM should be ready to accept connection requests.
Post-conditions	1. FNV2-CM has registered with FNV2-SOA and is ready to accept connection requests. 2. All the networking interfaces are accounted for and the status is known.
List of Exception Use Cases	Remote start via App
Interfaces	

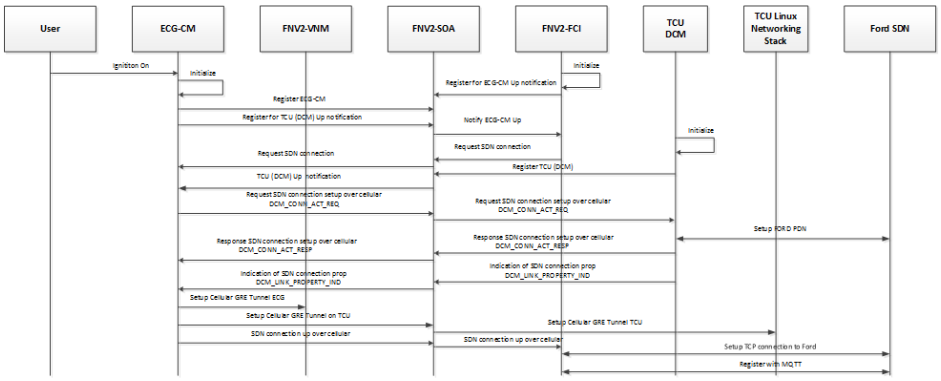

Figure 3

5.1.2. FTCP Connection Request

An application or service requests an FTCP Connection to Ford SDN. The request is handled by the FNV2-FCI and is accompanied by intent and FTCP payload. FNV2-FCI identifies the right transport based on FNV2-CM network interface data, and uses FNV2-CM to set up the route.

Actors	Application/service requesting FTCP Connection, FNV2-CM, FNV2-FCI, ECG-VNM
Pre-conditions	1. ECG, SYNC and TCU are in normal operation mode as opposed to sleep 2. TCU cellular connection and PDN connectivity is pre-established.

Actors	Application/service requesting FTCP Connection, FNV2-CM, FNV2-FCI, ECG-VNM
Scenario Description	1. App/service sends FTCP connection request to FNV2-FCI through FNV2-SOA. 2. FNV2-FCI sends request to FNV2-CM for FTCP connection with Ford SDN with payload and intent 3. FNV2-CM authenticates the request via FNV2-SecM 4. FNV2-CM sends ack to FNV2-FCI 5. FNV2-FCI queries the FNV2-CM for network interface list and respective statistics 6. FNV2-CM: 1. responds with a cached interface list (if that information is current and valid) OR 2. if needed: -triggers a status query to the various interfaces (TCU_CELL, TCU_WIFI, SYNC_WIFI etc) -relays the response of that query back to FNV2-FCI. 7. FNV2-FCI selects a route based on the intent and the statistics 8. FNV2-CM either dynamically sets up or assigns currently active GRE tunnel based on the route selected (interface with FNV2-L3-VNM) 9. FNV2-CM will set up the route (via either the TCU CM client or SYNC CM client) 10. FNV2-CM provides the status of the route back to FNV2-FCI 11. FNV2-CM adds the UID of the application/service to the “current connections” table
Post-conditions	1. A valid FTCP connection is established between the application/service and the Ford SDN 2. The application/service UID is recorded in the FNV2-CM "current connections" table
List of Exception Use Cases	
Interfaces	

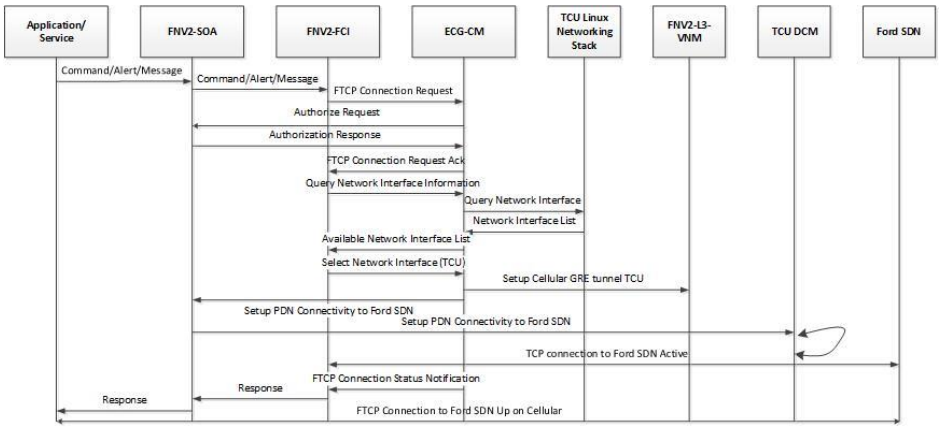

Figure 4

5.1.3. FTCP Connection Release Request

An application or service requests to release an existing FTCP Connection to Ford SDN. The request is handled by the FNV2-FCI and is relayed to the FNV2-CM. The assumption is that the connection is established over Cellular on TCU.

Actors	Application/service requesting release of FTCP Connection, FNV2-CM, FNV2-FCI, Networking Stack/Interface Controller
Pre-conditions	1. ECG, SYNC and TCU are in normal operation mode as opposed to sleep 2. TCU cellular connection and PDN connectivity is pre-established.
Scenario Description	1. App/service sends FTCP connection release request to FNV2-FCI through FNV2-SOA. 2. FNV2-FCI relays the request to FNV2-CM for release of the FTCP connection with Ford SDN 3. FNV2-CM authenticates the request via FNV2-SOA 4. FNV2-CM gracefully releases the connection over the active network interface 5. The release request response is relayed back to application/service via FNV2-FCI. 6. FNV2-CM removes the UID of the application/service from the “current connections” table

Actors	Application/service requesting release of FTCP Connection, FNV2-CM, FNV2-FCI, Networking Stack/Interface Controller
Post-conditions	1. FTCP connection between the application/service and the Ford SDN is closed 2. The application/service UID is removed from the FNV2-CM "current connections" table
List of Exception Use Cases	
Interfaces	

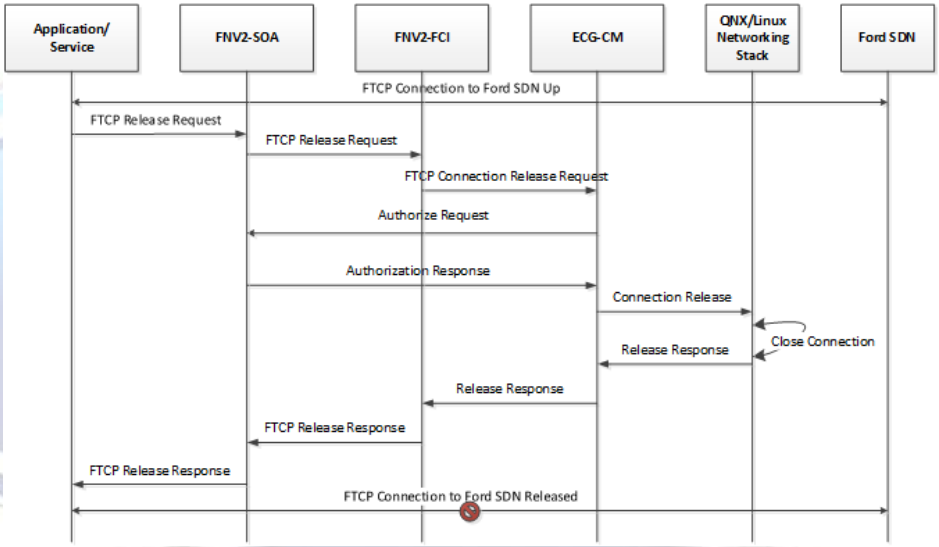

Figure 5

5.1.4. ECALL Handling

When an eCall is initiated over the TCU Cellular connection, it is mandated that the active data connection to Ford SDN over Cellular be closed and torn down. Once the eCall is completed, the cellular connection to the SDN needs to be re-established.

Actors	TCU ECALL service, FNV2-CM, FNV2-FCI, Networking Stack/Interface Controller, FNV2-SOA, TCU-DCM, TCU ECALL, Ford SDN
Pre-conditions	1. ECG and TCU are in normal operation mode as opposed to sleep 2. FNV2-CM is subscribed to the eCall related topics on FNV2-SOA 3. TCU cellular connection and PDN connectivity is pre-established.

Actors	TCU ECALL service, FNV2-CM, FNV2-FCI, Networking Stack/Interface Controller, FNV2-SOA, TCU-DCM, TCU ECALL, Ford SDN
Scenario Description	1. TCU ECALL service sets up eCall. 2. FNV2-CM is notified of the eCall via FNV2-SOA. 3. FNV2-CM gracefully releases active cellular data connection. 4. TCU ECALL completes the eCall. 5. FNV2-FCI initiates and re-establishes the connection to Ford SDN via FNV2-CM.
Post-conditions	1. No active cellular data connection during the eCall. 2. After the eCall cellular data connection is re-established.
List of Exception Use Cases	
Interfaces	

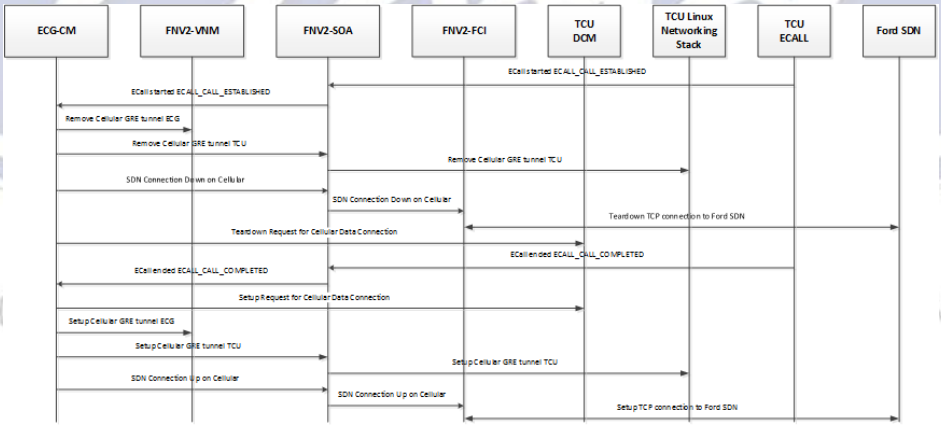

Figure 6

5.1.5. Offpeak Related Use Cases

5.1.5.1. Scheduling Offpeak

Actors	TCU ECALL service, FNV2-CM, FNV2-FCI, Networking Stack/Interface Controller, FNV2-SOA, TCU-DCM, TCU ECALL, Ford SDN
Pre-conditions	1. ECG, SYNC and TCU are in normal operation mode as opposed to sleep 2. TCU cellular connection and PDN connectivity is pre-established.
Scenario Description	1. Application sends streaming data connection request to FNV2-FCI through FNV2-SOA. 2. FNV2-FCI sends request to FNV2-CM for streaming data connection with the third party server. 3. FNV2-CM authenticates the request via FNV2-SecM 4. FNV2-CM sends ack to FNV2-FCI 5. FNV2-FCI queries the FNV2-CM for network interface list and respective statistics 6. FNV2-CM: 1. responds with a cached interface list (if that information is current and valid) OR 2. if needed: -triggers a status query to the various interfaces (TCU_CELL, TCU_WIFI, SYNC_WIFI etc) -relays the response of that query back to FNV2-FCI. 7. FNV2-FCI selects a route based on the intent and the statistics 8. FNV2-CM either dynamically sets up or assigns currently active GRE tunnel based on the route selected (interface with FNV2-L3-VNM) 9. FNV2-CM will set up the route (via either the TCU CM client or SYNC CM client) 10. FNV2-CM provides the status of the route back to FNV2-FCI 11. FNV2-CM adds the UID of the application/service to the “current connections” table
Preconditions	1. A valid FTCP connection is established between the application/service and the Ford SDN 2. The application/service UID is recorded in the FNV2-CM "current connections" table
List of Exception Use Cases	
Interfaces	

Figure 7

5.1.5.2. Start and Stop Offpeak

Actors	TCU ECALL service, FNV2-CM, FNV2-FCI, Networking Stack/Interface Controller, FNV2-SOA, TCU-DCM, TCU ECALL, Ford SDN
Pre-conditions	1. ECG, SYNC and TCU are in normal operation mode as opposed to sleep 2. TCU cellular connection and PDN connectivity is pre-established.
Scenario Description	1. Application sends streaming data connection request to FNV2-FCI through FNV2-SOA. 2. FNV2-FCI sends request to FNV2-CM for streaming data connection with the third party server. 3. FNV2-CM authenticates the request via FNV2-SecM 4. FNV2-CM sends ack to FNV2-FCI 5. FNV2-FCI queries the FNV2-CM for network interface list and respective statistics 6. FNV2-CM: 1. responds with a cached interface list (if that information is current and valid) OR 2. if needed: -triggers a status query to the various interfaces (TCU_CELL, TCU_WIFI, SYNC_WIFI etc) -relays the response of that query back to FNV2-FCI. 7. FNV2-FCI selects a route based on the intent and the statistics 8. FNV2-CM either dynamically sets up or assigns currently active GRE tunnel based on the route selected (interface with FNV2-L3-VNM) 9. FNV2-CM will set up the route (via either the TCU CM client or SYNC CM client) 10. FNV2-CM provides the status of the route back to FNV2-FCI 11. FNV2-CM adds the UID of the application/service to the “current connections” table
Preconditions	1. A valid FTCP connection is established between the application/service and the Ford SDN 2. The application/service UID is recorded in the FNV2-CM "current connections" table

Actors	TCU ECALL service, FNV2-CM, FNV2-FCI, Networking Stack/Interface Controller, FNV2-SOA, TCU-DCM, TCU ECALL, Ford SDN
List of Exception Use Cases	
Interfaces	

Figure 8

5.1.6. Streaming Application Connecting to a 3rd party Server

When an Application requests a connection for streaming media/data from a third party server, the use case is similar to a regular FTCP/HTTP connection request, except the fact that the application needs to be authenticated by the SDN Policy Manager module before the connection to the third party server can be established.

Actors	TCU ECALL service, FNV2-CM, FNV2-FCI, Networking Stack/Interface Controller, FNV2-SOA, TCU-DCM, TCU ECALL, Ford SDN
Pre-conditions	1. ECG, SYNC and TCU are in normal operation mode as opposed to sleep 2. TCU cellular connection and PDN connectivity is pre-established.

Actors	TCU ECALL service, FNV2-CM, FNV2-FCI, Networking Stack/Interface Controller, FNV2-SOA, TCU-DCM, TCU ECALL, Ford SDN
Scenario Description	1. Application sends streaming data connection request to FNV2-FCI through FNV2-SOA. 2. FNV2-FCI sends request to FNV2-CM for streaming data connection with the third party server. 3. FNV2-CM authenticates the request via FNV2-SecM 4. FNV2-CM sends ack to FNV2-FCI 5. FNV2-FCI queries the FNV2-CM for network interface list and respective statistics 6. FNV2-CM: 1. responds with a cached interface list (if that information is current and valid) OR 2. if needed: -triggers a status query to the various interfaces (TCU_CELL, TCU_WIFI, SYNC_WIFI etc) -relays the response of that query back to FNV2-FCI. 7. FNV2-FCI selects a route based on the intent and the statistics 8. FNV2-CM either dynamically sets up or assigns currently active GRE tunnel based on the route selected (interface with FNV2-L3-VNM) 9. FNV2-CM will set up the route (via either the TCU CM client or SYNC CM client) 10. FNV2-CM provides the status of the route back to FNV2-FCI 11. FNV2-CM adds the UID of the application/service to the “current connections” table
Preconditions	1. A valid FTCP connection is established between the application/service and the Ford SDN 2. The application/service UID is recorded in the FNV2-CM "current connections" table
List of Exception Use Cases	
Interfaces	

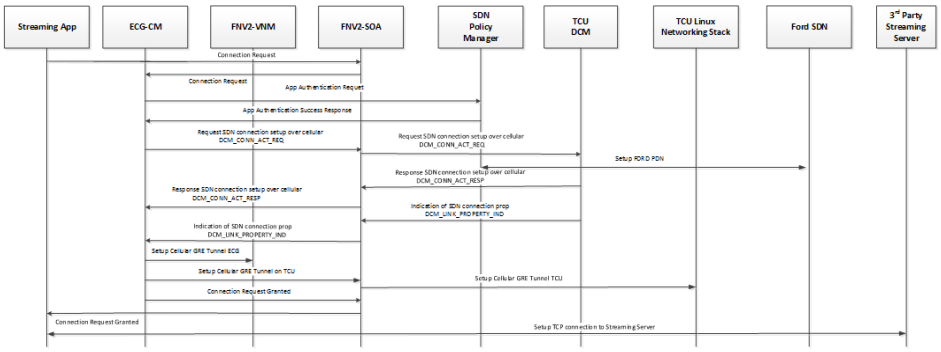

Figure 9

5.2. Component View

5.2.1. FNV2-CM

Connection interfaces - The Apps interact with the FNV2-CM through the Component "Connectivity Manager API". CM API's are provide on all the ethernet connected nodes ECG/TCU and SYNC. The connections states are managed by "Connection State Machines". FNV2-CM communicates with the interface managers (Cellular, WiFI, USB and Bluetooth) using FNV2-SOA framework.

Network Stack Configuration - FNV2-CM configures the GRE tunnel (Service and Client) on ECG/SYNC/TCU using the "Network Stack Configuration" component. The "Network Stack Configuration" components communicates with FNV2-VNM to setup the GRE tunnels and and route tables to bridge the GRE tunnels.

Network Statistics - FNV2-CM maintains per interface statistics in the "Traffic Stats" component.

Connection Authorization - FNV2-CM authorizes connection using "Connection Authorization" component, connection authorization is done on per App basis based on UID of the App. A pre-configured ACL is used to authorize App traffic. Apps can not use an IP interface until they have been authorized. At system boot up packet filtering rules will block all the traffic on all the nodes. Once App is authorized and it requests a connection, packet filtering rules will be added to allow traffic originating from an App to be forwarded to the desired IP interface. FNV2-CM will request FNV2-VNM to setup the packet filtering rules.

Connection Authentication - FNV2-CM authenticates the Apps by using the API's provided by Security Manager (TBD).

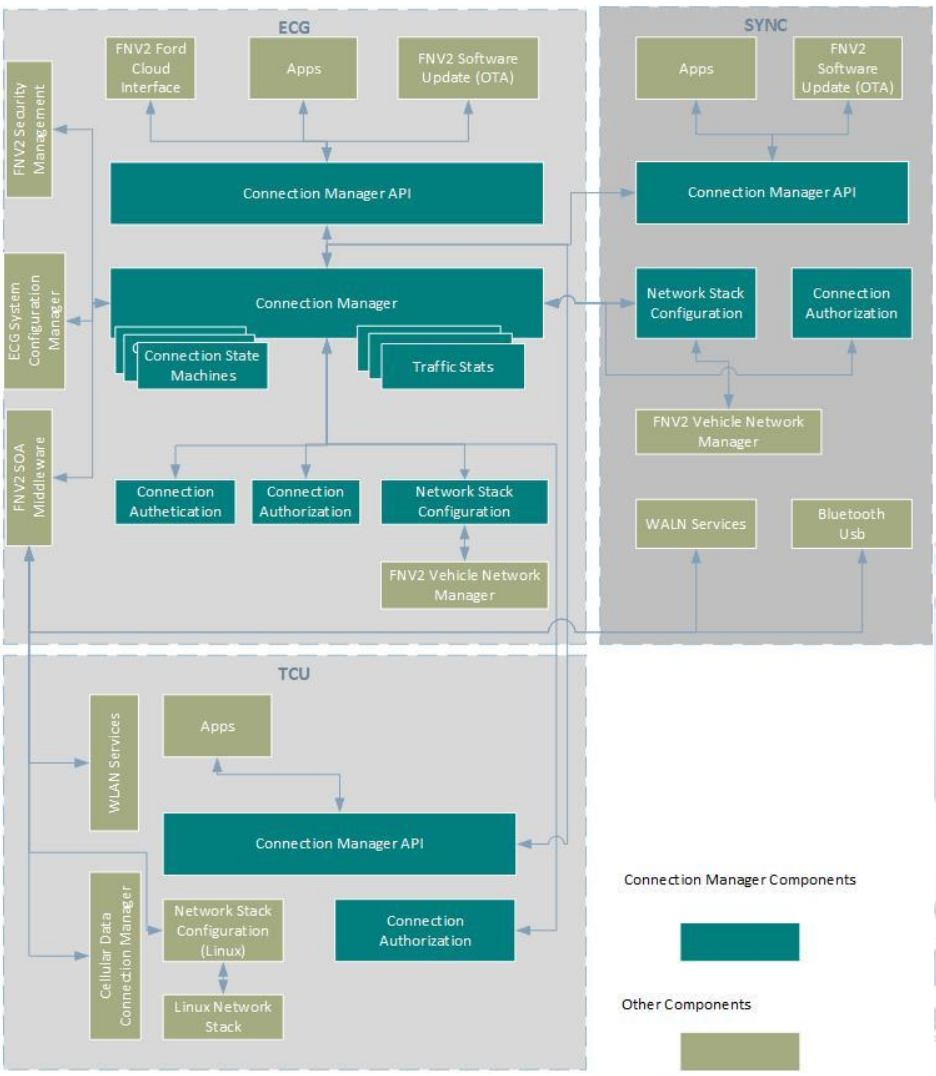

Figure 10

5.2.2. Connection State Machine

The networking connection interfaces will use the following state machine. At boot up each networking interface will be in "NULL" state. After an interface is initialized it will be put into disconnected state. The state machine below shows the various state transitions depending on incoming event. The state machine in connected state implies that the interface is up with an IP address. The state machine in the online state implies there is a route to the Ford SDN. FNV2-CM will perform and perform a connectivity test on the each interface that is in the connected state before publishing it as online. Apps should only use the connections that are marked online i.e. has full connectivity to network. Network transport interfaces types covered by the state machine are Cellular,WiFi, Bluetooth and USB Connection.

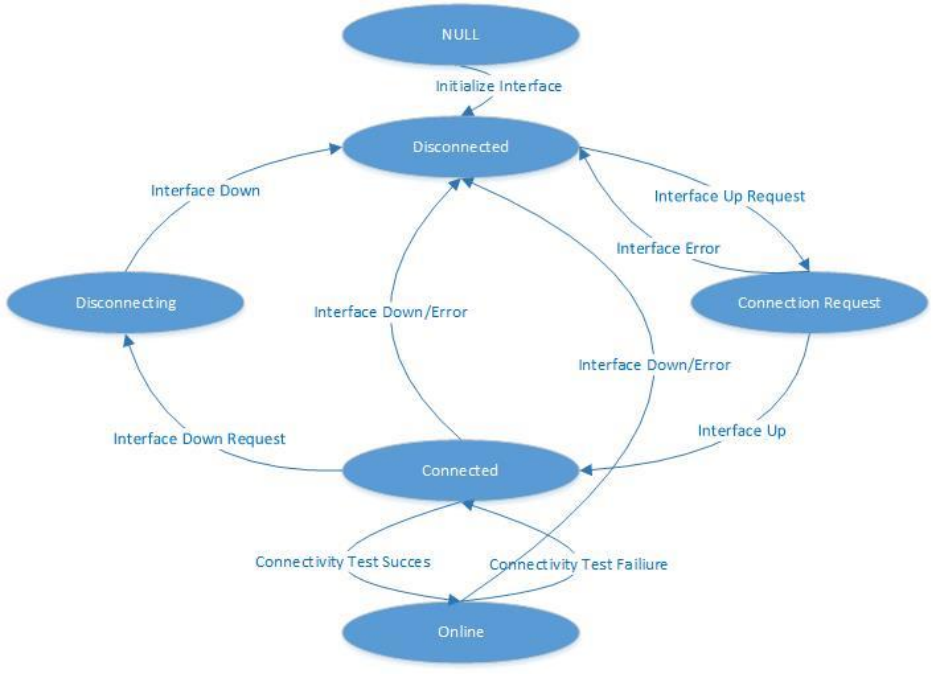

Figure 11

5.2.3. Network Interface Virtualization Management

5.2.3.1. VLAN Management

Virtual local area network (Vlan) is a logical grouping of devices that shares the same network regardless of the physical location of said devices. Vlan can be used to partition a network on the same physical location into multiple independent networks as well as to form a network from devices that are situated in separate locations.

FNV2-CM establishes Vlans across all ECU so that any applications will have access to all available network interfaces as if they were local to the ECU. This mechanism also provides a method for applications to share networks with the same priority level as well as enabling applications to seamlessly switch among network interfaces based on availability, QoS, and priority.

When establishing a Vlan, FNV-CM will try to create two Vlan nodes – a host node that resides on the same ECU as the requested application, and a gateway node that resides on the ECU where the network interface exists. Once both nodes are created and linked, an IP address of the host node is return to the application. Any data sent to the host node will be rerouted to the gateway node, and finally delivered to the network interface.

5.2.3.2. GRE Tunnels

The FNV2-CM is responsible for authenticating, authorization and setting up the data path through the ECG (IP layer), for the applications that are requesting a connection. Since both

Sync and TCU have multiple network interfaces present on them, there is a need to separate the traffic on to a specific interfaces depending on the the application request. FNV2-CM will use GRE tunnels between ehternet nodes to map traffic to a specific interface, as requested by the application. FNV2-CM will map each network interface on the TCU to a separate GRE tunnel. If there are 3 outgoing interfaces on the TCU then each outgoing interface will have a corresponding GRE tunnel. The procedure is similar for SYNC network interfaces.

For example if on the TCU there are 3 outgoing interfaces present (Ford APN, Internet APN and WLAN). For each of these outgoing interface on the TCU, CM will create a service GRE tunnel between ECG and TCU, so that the application traffic can routed to the correct outgoing interface on TCU. For traffic originating on ECG; if the traffic is to be directed the FORD APN, the traffic has to be mapped to the "GRE Ford APN" tunnel.

For traffic originating from SYNC that needs to go out over TCU, there is need to separate the incoming traffic from SYNC based on the target interface the traffic needs to go over. Here we use GRE tunnels again to separate the incoming traffic into ECG. These GRE tunnels are referred to as client GRE tunnels. Traffic originating on Sync first goes over a client GRE tunnel to ECG, where it is then mapped to a Service GRE tunnel going out to the TCU. The figure below shows the mapping of different service and client GRE tunnels on SYNC, ECG and TCU and other Ethernet connected modules. Connectivity Manager in concert with "Network Stack Configuration" component will setup these GRE tunnels for an application requesting a connection.

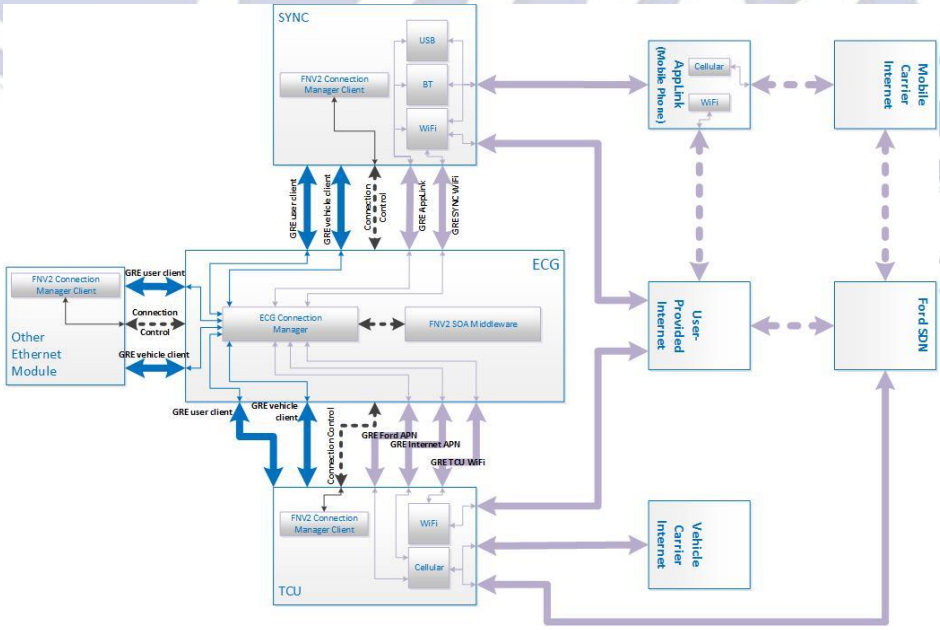

Figure 12

To configure a GRE tunnel the FNV2-CM has to configure the networking stack on both end of the GRE tunnel. The example below shows the parameters that are needed to configure a GRE tunnel between ECG and TCU.

5.2.3.2.1. ECG Networking Stack

```
GRE Interface Name: TunCellular1
Associated Outgoing Interface: Ford SDN APN name
Local IP Address: 10.0.1.3
Remote IP Address: 10.0.1.4
Tunnel Source: 192.168.1.2
Tunnel Denstination: 192.168.1.3
IP MTU : 1400
TCP MSS: 1360
```

5.2.3.2.2. TCU Networking Stack

```
GRE Interface Name: TunCellular1
Associated Outgoing Interface: Ford SDN APN name
Local IP Address: 10.0.1.4
Remote IP Address: 10.0.1.3
Tunnel Source: 192.168.1.3
Tunnel Denstination: 192.168.1.2
IP MTU : 1400
TCP MSS: 1360
```

5.2.4. Provisioning

Provisioning client uses API's provided FCI for connecting to Ford cloud during provisioning. FCI uses FNV2-CM API's to manage bring up of cellular or WiFi interface based on request from the provisioning client. For bring up of WiFi interface FCI provides SSID and password.

FNV2-CM is not responsible for configuring, maintaining or updating Cellular APN's. Cellular APN's are pre-configured at the factory or provided through SMS message.

FNV2-CM is not responsible for configuring maintaining or updating WiFi profiles used for provisioning.

6. Connectivity Policy, Diagnostics, and Configuration Files

6.1.1. Intents

Applications or system services request CM for wireless network interfaces of a given type (Cellular or WiFi) for cloud/internet connectivity. They use a messaging construct called **Intent** to describe preferred interface, expected time of service, applicable fail-over rules, priority, and

expiry of a request. To fulfill the requests, depending on the lifetime of Intents, certain type of **Intents** (eg. Off-Peak Intents) must be retained post ECG reboots or ignition cycles.

Those **Intents** are stored in *data* partition on all ECUs hosting WIR applications or system services. The information persists over a software upgrade or downgrade and is subject to erasure at factory reset.

ECU	File Path
ECG	/data/config/cm/cmIntents
TCU	/data/cm/cmIntents
SYNC4	/fs/storage/rwdata/cm/config/cmintents

In the event of file corruption, CM would replace the file with a backup copy stored in *data* partition.

6.1.2. Intents Policy Table

Ford cloud manages unique application ID and wireless interface privileges of each WIR application or system service for cloud/internet connectivity. CM identifies each WIR application by the application ID. When receiving Intent requests from an application, CM validates the application against its Intent privileges.

Intent Policy is stored in a permanent partition on all ECUs hosting WIR applications or system services. The table persists over a software upgrade or downgrade and is not subject to erasure at factory reset.

At production, CM stores the default policy of all those managed applications in *perm* partition on ECG. Likewise, it stores them in *data* partition on TCU and SYNC4. Whenever there is a change to the policy table, cloud pushes it OTA to vehicles for update. Upon receipt, CM replaces the table with the new one on ECG followed by its peer ECUs (eg. TCU, SYNC4). In addition, CM sends update status (success or failure) to Ford cloud that would resend the version of the table that failed until the update is successful.

ECU	File Path	Comments
ECG	/perm/config/cm/cmpolicy	In fully connected operation
TCU	/data/cm/cmpolicy	In fully connected operation
SYNC4	/fs/storage/rwdata/cm/config/cmpolicy	In fully connected operation

In the event of policy file corruption, CM would try to secure a copy of the table from its peer ECU(s) to replace the corrupted file. If this attempt fails, CM would replace the policy file with default policy and immediately reports the condition to cloud to receive an OTA update.

6.1.3. WIR Configuration DIDs

Ford Cloud manages a set of WIR-specific, EOL-provisioned as well as OTA-updatable configuration DIDs.

Those DID values are stored in *data* partition on ECG. The information persists over a software upgrade or downgrade and is subject to erasure at factory reset.

ECU	File Path
ECG	/data/config/cm/cmConfigData

At production, CM obtain the default values of those DIDs from SPCM and saves them on ECG. Whenever those DIDs are updated OTA, SPCM pushes new values to CM which would in turn update the file.

In the event of file corruption, CM would replace the file with a backup copy stored in *data* partition. If that fails, CM would replace it with default values.

6.1.4. Network Interface Allocation History

To facilitate diagnosis of application connection failures, CM collects network interface allocation/release history from all ECUs and aggregates them on ECG. At the end of every 30 days, CM sends the information to Diagnostics portal.

Network Interface Allocation History is stored in *data* partition on ECG. The information persists over a software upgrade or downgrade and is subject to erasure at factory reset.

ECU	File Path	Comments
ECG	/data/config/cm/nwIfAllocHistory.log	In fully connected operation
TCU	/data/cm/nwIfAllocHistory.log	Only if in standalone operation
SYNC4	/fs/storage/rwdata/cm/config/nwIfAllocHistory.log	Only if in standalone operation

In the event of file corruption, data loss is expected. CM would not implement any backup strategies. CM would delete the corrupted file and continue collecting new history.

6.1.5. WLAN Diagnostics

To facilitate diagnosis of WiFi interface disconnections in the vehicle, each time an interface goes down, CM collects diagnostics information on the WiFi interface from respective WiFi edge ECU (TCU or SYNC4) and aggregates them on ECG. Upon request from cloud or at the end of every 30 days, CM sends the information to Diagnostics portal.

WiFi Diagnostics is stored in *data* partition on ECG and WiFi edge ECUs (i.e. TCU and SYCN4). The information persists over a software upgrade or downgrade and is subject to erasure at factory reset.

ECU	File Path	Comments
ECG	/data/config/cm/wlanIfConnHistory.log	In fully connected operation
TCU	/data/cm/wlanIfConnHistory.log	Only if in standalone operation
SYNC4	/fs/storage/rwdata/cm/config/wlanIfConnHistory.log	Only if in standalone operation

In the event of file corruption, data loss is expected. CM would not implement any backup strategies. CM would delete the corrupted file and continue collecting new history.

6.1.6. WLAN Profiles

CM scans available WLAN APs and presents the ones with good signal strength to SYNC4 HMI. To connect to an AP, user would enter a password. CM saves SSIDs of already visited APs along with their passwords called WLAN Profiles. In addition, CM saves TCU WHS AP credentials to the profiles.

WLAN Profiles are stored in *data* partition on all ECUs. The information persists over a software upgrade or downgrade and is subject to erasure at factory reset.

ECU	File Path
ECG	/data/config/cm/wlanProfiles
TCU	/data/cm/wlanProfiles
SYNC4	/fs/storage/rwdata/cm/config/wlanProfiles

In the event of file corruption, CM would replace the file with a backup copy stored in *data* partition.

7. Constraints and Dependencies

FNV2-CM is dependent on the existence of a security framework for authenticating and authorizing applications and services that may generate connection requests. Currently, in FNV2-SOA MQTT framework, there is no per application or service support. A common security framework for authenticating the applications and services is an area that is currently under active investigation.

8. Future Considerations

8.1.1. MPTCP

MultiPath TCP (MPTCP) is an effort that provides "better resource utilization, better throughput and smoother reaction to failure". In the context of FNV2-CM, MPTCP is a means of providing lowest cost route and for switching between multiple interfaces without bringing down an established TCP session with the Ford backend. The fastest and cheapest subflow is used (if multiple subflows/interfaces present) and the other subflows are retained as open back-up mechanisms.

Figure 13 provides a view of the connectivity of an MPTCP enabled system. With MPTCP, WiFi and Cellular interfaces on a module can be utilized for handoff, bandwidth aggregation and redundancy.

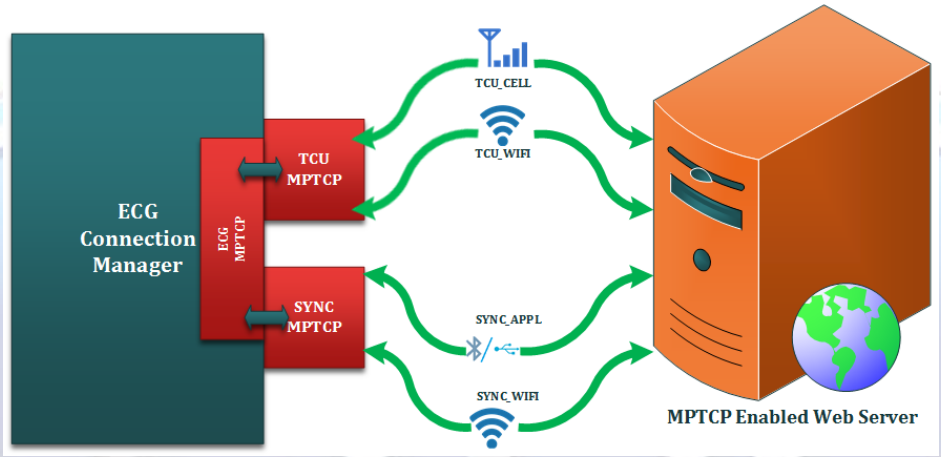

Figure 14

Provided in Figure 14 is an overview of MPTCP network stack.

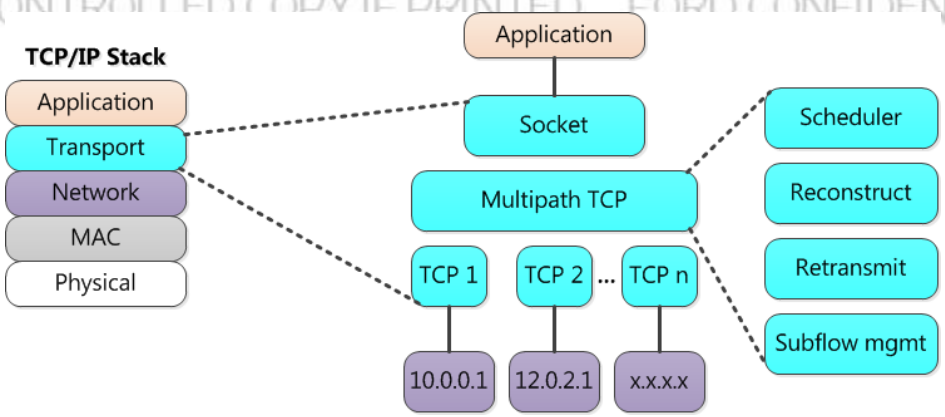

Figure 15

The availability of MPTCP as a feature for FNV2-CM is contingent on commercial implementation in QNX 7.0 (for ECG and SYNC) and MPTCP support on the server side. **Hence, as of now, this feature is slated for future consideration.**

Possible future provisioning items for FNV2-CM include MPTCP stack and its related parameters.

9. References

Documents
Connectivity Manger Overview.pptx
IVDCM feature Specific Use Cases.docx
IVDCM.SAD-V.60_Draft.pdf
FNV2 Connection Manager Workshop.pptx
Hardware System_Architecture Specification_20161207.docx