ECG Connection Manager (ECG-CM) Detail Design

Status	IN PROGRESS
Completion (%)	10%
DD Version	0.5
HLD Name	FNV2 Connection Manager Architecture (FNV2-CM)
HLD Version	0.3
JIRA	ECG-2687
Owner	Brar, Karmindar (K.) Marupaka, Nagendra (N.) Park, Seung (S.D.) Unknown User (memborg) Mandya Nayak, Prakash (P.)
Reviewer	Lisak, Darko (D.) Murugesapillai, Nava (N.) Unknown User (amiramo3) D'Souza, Stephen (S.) Puri, Ron (R.) Ma, David (D.) Lopez, Richard (R.) Kirstein, David (D.)
Review Date	v0.1 - 27 Jun 2017
Approver	Murugesapillai, Nava (N.)
Approve Date	v0.1 - 05 Jul 2017

1. Introduction

ECG Connection Manager (ECG-CM) enables any and all in-vehicle applications or services to use network interfaces on any FNV2 Ethernet connected ECU for cloud connectivity. The network interface can be on the same ECU as the application or on a different ECU. Typically, the following factors need to be evaluated when cloud connectivity is requested:

- Urgency, priority and time-criticality of the request
- Endpoint being requested (whether it is within Ford's cloud or on a third party infrastructure)
- Status of the network interfaces currently available and the associated cost, load and reliability

Applications/services communicate the first two factors to the ECG-CM by providing an Intent as part of the cloud connectivity request. Subsequently ECG-CM evaluates the intent and the network interface status to formulate a policy for processing the request.

ECG-CM:

1. Processes and keeps track of all cloud connectivity requests from applications/services (these requests are processed, discarded or queued).
2. Interacts with and maintains the states of all the network interfaces on all Ethernet connected ECUs. They include: TCU DCM (Cellular modem), TCU WLAN, SYNC WLAN and SYNC AppLink.
3. Brings up, maintains and tears down networking components such as GRE tunnels, routes and linkages to network interfaces. This includes GRE tunnel ID and IP address assignment functionality. ECG-CM achieves this via interaction with the FNV-L3-VNM components on the ECG, SYNC and TCU.
4. Has additional functionality requirements in special conditions such as Offpeak network traffic, Emergency call and Provisioning mode.

This detailed design of ECG-CM has been formulated based on the above high-level functionalities and requirements.

The subsequent sections of this document provide detailed description of the components, messaging mechanisms, class definitions and data structures employed within ECG-CM.

1.1. Nomenclature: WIR vs CM

The term Wireless Interface Router (WIR) encompasses the broader logic of cloud connectivity and the related policy management, analytics and diagnostics across all ECU's. For this document's purposes, the term WIR and CM will be used interchangeably unless explicitly stated otherwise. The nomenclature of WIR shall be employed in the context of applications/services that will be requesting the cloud connectivity. The term ECG-CM will be employed when the context is the architecture of the Connection Manager.

A cross-platform library of APIs will be available for applications/services on any and all ECUs to leverage the services provided by ECG-CM. Henceforth, this library will be referred to as WIR API Library. Consequently, any application/service that requests cloud connectivity services from ECG-CM will be henceforth referred to as WIR Client. The APIs are documented here and the programmer's guide is provided here.

It is to be noted that since WIR Clients will be creating and maintaining their own sockets and data transactions, ECG-CM is essentially allocating a suitable network interface (not a connection) over which a WIR Client can establish data transport. Henceforth, cloud connectivity requests will be referred to as network interface allocation request or allocation request (in short).

2. Component Overview

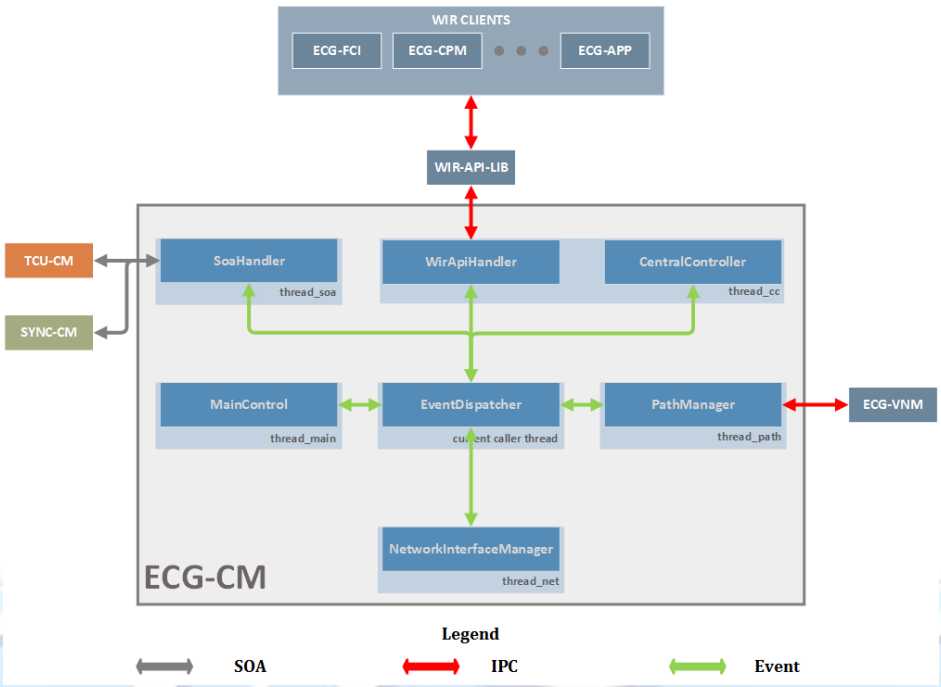

Figure 1: Component Overview

Figure 1 illustrates the components present in ECG-CM and their external and internal interactions.

MainControl is the main singleton class that is instantiated by the ECG-CM process and is composed of various member classes.

WirApiHandler: interfaces with the WIR API Library to receive the incoming allocation requests, callbacks and notifications from WIR Clients. Subsequently, these requests and notifications are relayed to CentralController for processing.

CentralController: processes the allocation request from the WIR Clients and sends either an immediate request to MainControl to bring up an interface or it queues the request on itself to be serviced when certain conditions are met i.e. when a particular interface becomes available, scheduled downloads, low cost route (WLAN). The CentralController provides the MainControl the routeData when a request is made, along with the AllocId. AllocId is unique to for every request that comes in from an WIR Client.

PathManager : is responsible for setting up the data flow path through FNV2 connected modules for an app the requests a connection. Pathmanager aggregates all data flow paths in the system- A path describes a data flow from WIR Client to an outgoing interface. A path is made up of a combination of tunnels and an outgoing interface. Some path may have a client tunnel and service tunnel, other may only have a service tunnel. The tunnels and interfaces are linked at IP layer (routing), to map the traffic from one tunnel to another and then to an outgoing interface.

NetworkInterfaceManager - an instance of NetworkInterfaceManager aggregates all the network interfaces in the system. An instance of an network interface aggregates the network interface state, network interface configuration and statistics associated with a network interface.

SoaHandler - This module is responsible for subscribing to all SOA topics that are of interest to the ECG-CM. Its also responsible for publishing all the SOA messages to the external WIR Clients. The SoaHandler is responsible for mapping SOA messages to internal events for all incoming SOA messages.

EventDispatcher - This class performs event handling duties. Class instances can subscribe to events of interest. The event dispatcher offers API's for publishing events and receiving notifications for events.

To bring up a route to an Interface the MainControl first queries the NetworkInterfaceManager to check if an interface is available. If an interface is not available a request is made to the Network interface bring up an interface. Once and Interface is available the MainControl requests the PathManager for a path to the Interface. The PathManager is responsible for bringing up the tunnels and linking the tunnels and Iface togather to provide a complete path for the WIR Client data to be routed.

3. Process and Threading Model

3.1. Process

ECG-CM runs in a single process on FNV2 ECG at start up and stays up as long as the ECG is powered. All the ECG-CM functionality is managed with in this process. ECG-CM communicates to other services and process on ECG and other Ethernet connected FNV2 ECU's using the SOA framework. The messaging is defined using Google Protocol Buffers.

3.2. Thread Model

On startup, the main ECG-CM thread creates a MainControl singleton instance running in the main thread. ECG-CM instantiates an instance of CentralController, an instance of PathManager, an instance of NetworkInterfaceManager, and a SOA handler. Each running in its own thread with a message queue to process the incoming events in the order of arrival. The queues are thread safe. Each event is processed to completion, before servicing the next event in the queue.

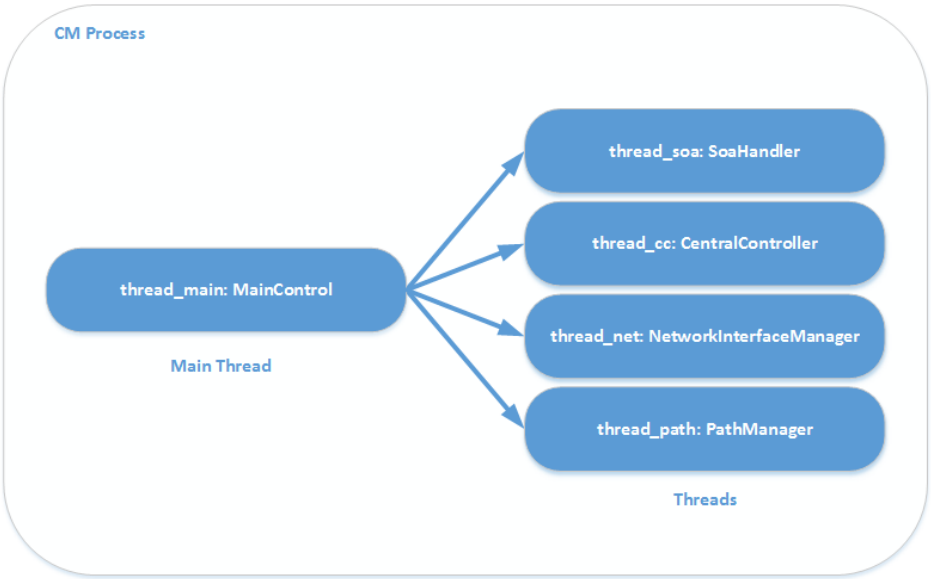

Figure 2: ECG-CM Process and Threads

3.2.1. Thread table

Thread	Classes	Description
thread_main	MainControl	This is the main thread responsible for the creation and execution of the MainControl singleton class. The MainControl in turn creates instances of CentralController, PathManager, NetworkInterfaceManager, and SoaHandler during its initialization and assigns these instances to separate threads.
thread_soa	SoaHandler	This is the SoaHandler thread which is responsible for registering with SOA framework. This thread is responsible for handling the publish and subscribe of the SOA messages from the ECG SOA framework. Upon receiving the SOA messages this thread maps them to internal events and publishes in to the EventDispatcher.
thread_cc	CentralController	The CentralController thread is responsible for handling WIR Client Intent processing and creating low level policy to provide the WIR Client the desired network interface. This thread uses the timers to schedule time day scheduling requests.

thread_net	NetworkInterfaceManager	This is the NetworkInterfaceManager thread, which is responsible for managing and monitoring all wireless interface on all Ethernet Connected ECUs's. In particular TCU Cellular through DCM, TCU WLAN STA, SYNC WLAN STA and SYNC AppLink.
thread_path	PathManager	This is the PathManager thread, which is responsible for setting up the routing path from an originating ECU to an outgoing interface. The PathManager setups up tunnels and link the outgoing interfaces to a tunnel using the FNV2 VNM on different ECU's.

3.2.2. Inter-thread Messaging

Messaging inside the ECG-CM is done using Events.The Observer design pattern is used for event handling. The event posting is done through the EventDispatcher class. All incoming SOA message are mapped into events, inter thread communication is also done by posting events. Each instance of a class subscribes to the events it is interested in handling through the subscription API's provided by the EventDispatcher class. As events are published form external and internal providers the EventDispatcher' notifies the registered observers of the event. The EventDispatcher runs in the caller thread and queues the Events on to observers queue. Each observer queue is thread safe and protected by mutex. An Event is allocated once, after that only reference to the event is passed and reference count is maintained using C++ construct, std::shared_ptr.

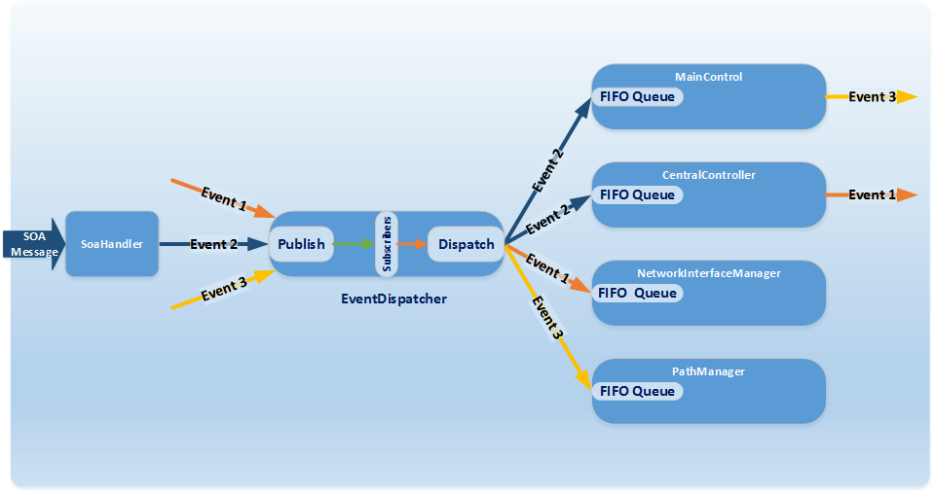

Figure 3: Event Queuing Example

4. Component Class Definitions

4.1. MainControl

MainControl singleton aggregates the CentralController, WirApiHandler, PathManager, NetworkInterfaceManager, and SoaHandler classes. MainControl maintains a list of Route Data. Route Data is unique set of Iface, originating ECU, Termination ECU, Bypass ECG, an associated priority and state, this set of data is identified with a unique RouteId. As WIR Clients issue their intents to request an interface, the WIR Clients are assigned a unique AllocId to track their requests. One RouteID may have many AllocId mapped to it, if the route requirements are identical.

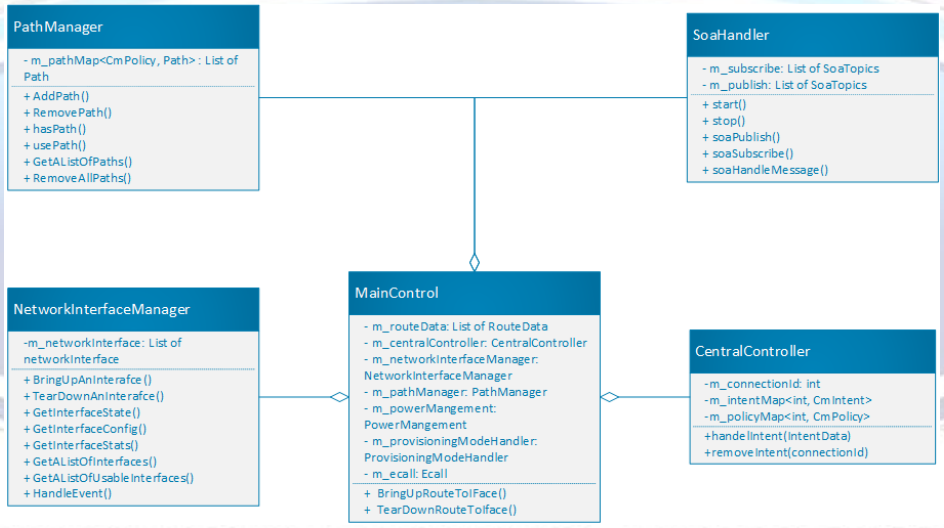

Figure 4: CM Main Control Singleton Class

```
typedef struct Route_Data {
    bool ready;
    IfaceType_t iface_type;
    ConnType_t type;
    bool bypass_ecg;
    EcuType_t req_ecu;
    EcuType_t iface_ecu;
    PriorityType_t route_priority;
} Route_Data _t;
```

4.2. WirApiHandler

WirApiHandler within ECG-CM is the component that interfaces with the WIR API Library to receive the incoming allocation requests, callbacks and notifications from WIR Clients.

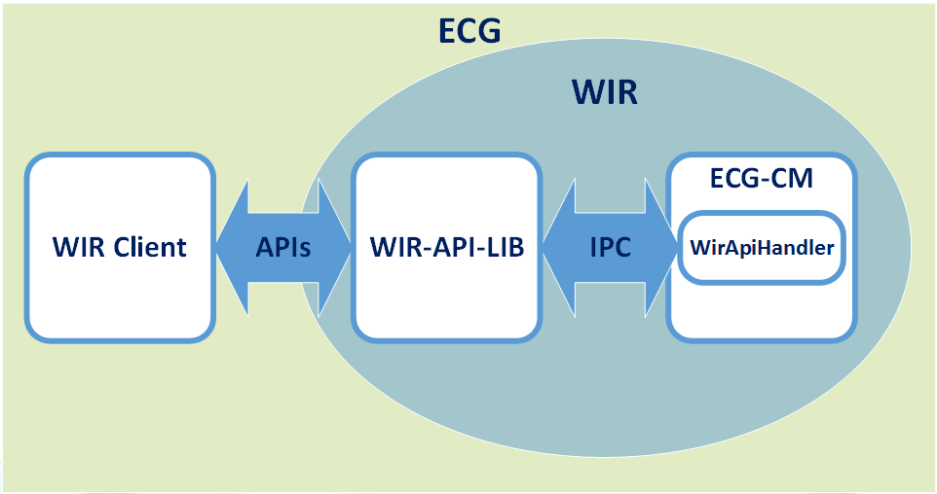

Figure 5: WirApiHandler

4.3. Central Controller

ECG-CM employs the concept of intent as a way for WIR Clients and services to communicate their need, priority and urgency for a cloud/internet connection. The CmIntent, CmPolicy and CentralController classes provide this ability.

- **CmIntent**: This class abstracts the intent that the WIR Client/service needs to be provide as part of the allocation request. The incoming SOA message with the allocation request (from FNV2-FCI, for instance) will include intent information in the payload that will be transformed and instantiated into a CmIntent object and enqueued in CmIntentList for eventual handling. Subsequently, an event for Intent Handling will be raised.
- **CmPolicy**: This class captures the low level policy information required to formulate the route and path for a specific allocation request with an intent.
- **CentralController**: This class encapsulates the CmIntent lists and the handling methods. When an IntentHandle event is raised, CentralController will employ a mechanism that would transform the current CmIntent object into a CmPolicy object that is mapped to a Path object in the PathManager.

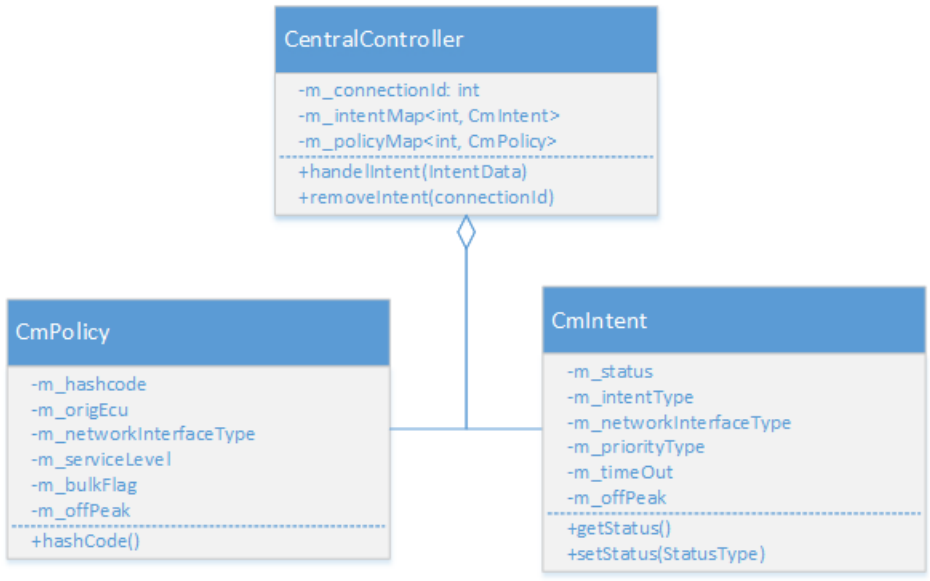

Figure 6: Central Controller Class

A rudimentary message flow is provided here to illustrate how the CentralController works with other CM components to achieve timely intent handling and responses to allocation requests.

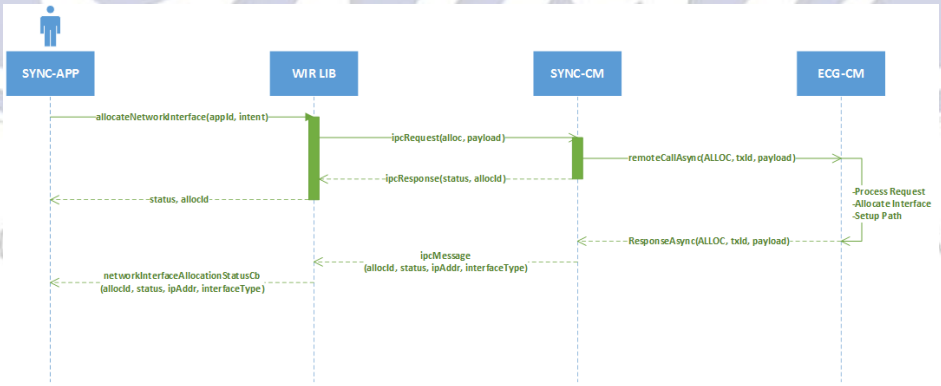

Figure 7: allocation request Sequence Diagram

4.4. Network Interface State Machine

The ECG-CM manages the Cellular, WLAN and AppLink interfaces on various ECU's. It uses the state machine described below to manage the state of each interface.

- **NetworkInterfaceManager** - Singleton class which aggregates all the network Interfaces

- **NetworkInterface** - An instance of an interface to be managed. This class aggregates the network interface state, network interface configuration and statistics associated with a network interface.
- **NetworkInterfaceState** - Abstract class for network interface state. This class is implemented by concrete classes, Disconnected, Connecting, Connected, Disconnecting, Initial and Online.

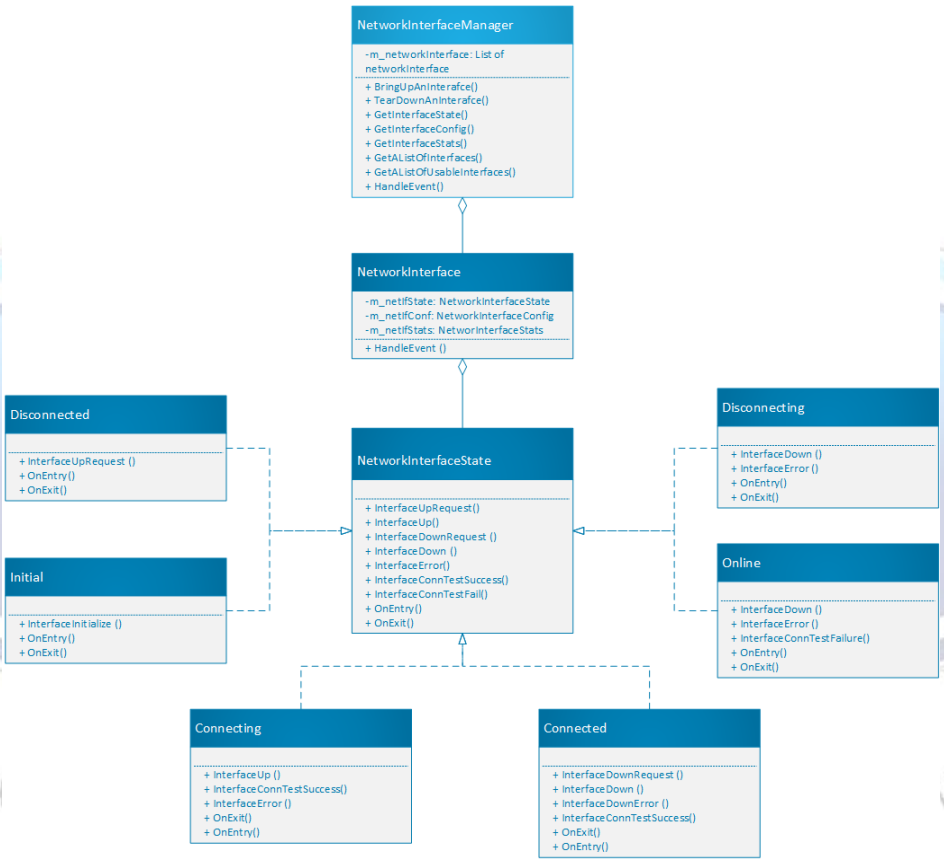

Figure 8: Network Interface State Machine Class

4.5. Network Path Manager

The ECG-CM manages the setup of networking path from source to destination ECU. This component essentially encapsulates the interaction of ECG-CM with the resident VNM component.

- **PathManager** - single instance of this class which aggregates all data flow paths in the system.
- Path - A path describes a data flow from WIR Client to an outgoing interface. A path is made up of a combination of tunnels and an outgoing interface. Some path may have a client tunnel and service tunnel, other may only have a service tunnel. The tunnels and

interfaces are linked at IP layer (routing), to map the traffic from one tunnel to another and then to an outgoing interface. There may be more than one path through the system based on service level of the path and an outgoing network interface.

- **NetworkInterface**- An instance of an interface to be managed. This class aggregates network interface state, network interface configuration and statistics associated with a network interface.
- **Tunnel -** A tunnel is a GRE tunnel that is created between 2 ECU's. Tunnels are created to separate the traffic based on outgoing interfaces.
- **TunnelEndpoint** - A tunnel creation involves setting up the tunnel endpoints on the 2 ECU's between which the tunnel is created. The tunnel endpoint defines tunnel IP address on the local ECU, the local IP address of the interface its bound to and the destination IP address of the interface where packets are forwarded. There is a unique Tunnel ID associated with each tunnel, referred to as kernelId. If more then one tunnel is created between the same source and destination ECU's, kernel key is used to separate the traffic flow between those tunnels.
- **IfaceLink** - Once the tunnels and interfaces are brought up they need to be linked together so the data packets can be forwarded from one tunnel iface to another tunnel iface and from a tunnel iface to an outgoing iface. The IfaceLink class is responsible for the linking and unlinking operations of the ifaces present in Path.

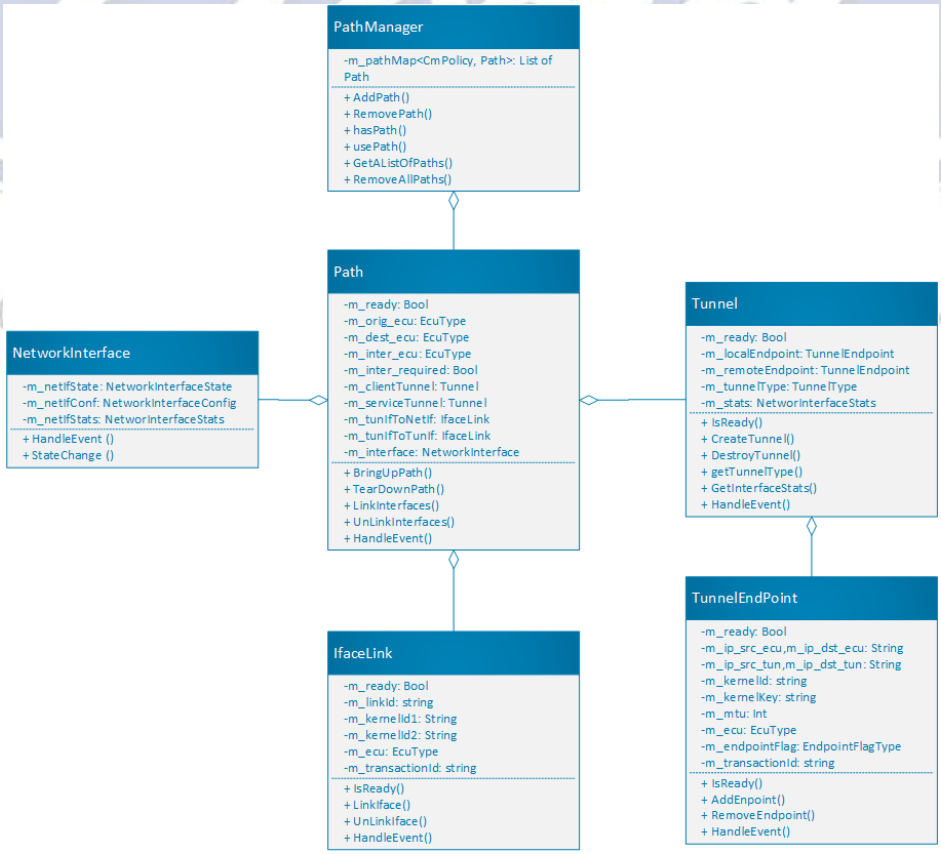

Figure 9: Network Path Manager Class

4.6. SOA Handler

The SoaHandler class implements the SOA interface employed by ECG-CM to communicate with inter-ECU CM entities like TCU-CM or SYNC-CM. This component consists of an instance of SoaProvider and SoaConsumer to cater to the various request response messages that occur between the various CM entities.

4.7. Event Dispatcher

All incoming SOA messages or IPC calls will be mapped into an in event. The events are separated into ECG-VNM, TCU DCM, WLAN services, AppLink and allocation request. The events will be forwarded to the interested classes by the EventDispatcher.
The event dispatching follows the observe/listener pattern. In the figure below, NetworkInterface class subclasses from the Listener class. The NetworkInterface class add itself as a listener on the EventDispatcher for NetworkInterfaceEvents.

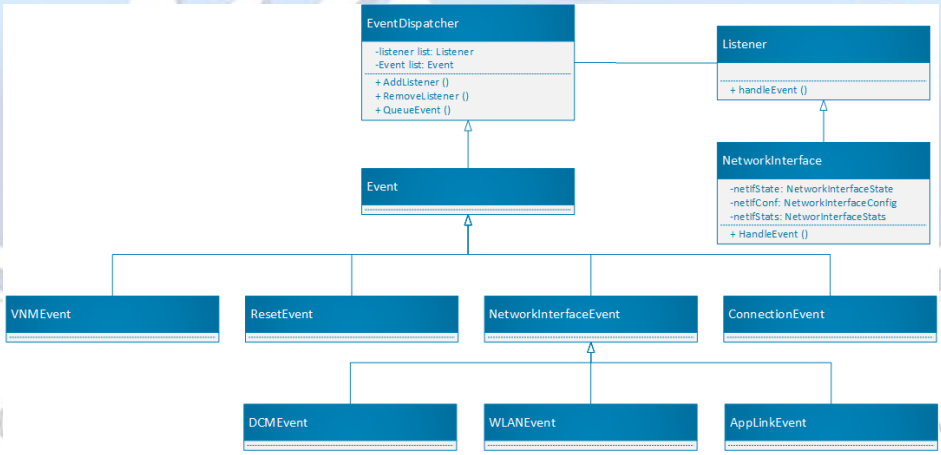

Figure 10: Event Dispatcher Class

4.7.1. TCU Cellular DCM

The NetworkInterfaceManager creates an NetworkInterface to manage the bring up and life cycle of a cellular interface. There may be more then one cellular interface, each managed through its own NetworkInterface instance. A cellular interfaces are defined by the APN's they map. A logical APN name mapping is created for all the APN's controlled by the TCU cellular DCM. Different logical APN name may be mapped to one actual APN or a unique APN. The logical APN names that are defined between ECG-CM and TCU DCM are INTERNET, FORD,TETHERING, FOTA, and HTTP. A NetworkInterface is responsible for bring up a one logical APN.384

The interaction between the NetworkInterface instance and the TCU DCM cellular is mapped to the following events.

4.7.1.1. DCM Events

Event	Description
DcmConnActivateReqEvent	event triggers bring up of a cellular APN
DcmConnActivateRespEvent	event informs if cellular APN bring up was a success or failure
DcmConnDeactReqEvent	event triggers tear down of the cellular APN
DcmConnDeactRespEvent	event informs if tear down cellular APN was success and failure
DcmConnDeactIndEvent	event to indicates when cellular interface goes down
DcmConnActIndEvent	event to indicate when cellular interface comes up
DcmLinkPropIndEvent	event supplies link property info like, ifname, DNS server IP addresses, MTU
DcmConnResetEvent	event to trigger clean up of all cellular connections on TCU DCM
DcmSessionInfoEvent	event to track current DCM session, if event session different then last, all DCM request need to be reissued

4.7.2. WLAN Services

The NetworkInterfaceManager creates an NetworkInterface to manage the bring up and life cycle of a WLAN interface on TCU and another instance to manage the WLAN interface on the SYNC. The TCU WLAN can be put in STA mode when required. SYNC WLAN always operates in STA mode.

The interaction between the NetworkInterface instance and the WLAN services is mapped to the following events.

4.7.2.1. WLAN Events

Event	Description
WlanStaEnableReqEvent	event triggers the activation STA mode on TCU WLAN
WlanStaEnableRespEvent	event informs if the activation of STA mode was started, success or failure on TCU WLAN
WlanStaDisableReqEvent	event triggers the deactivation STA mode on TCU WLAN
WlanStaDisableRespEvent	event informs if the deactivation of STA mode was started success or failure on TCU WLAN
WlanScanReqEvent	event triggers a scan on WLAN module
WlanScanRespEvent	event informs if scan is in progress
WlanScanDataEvent	event informs the results of the scan from WLAN module
WlanConnectReqEvent	event triggers connect to an AP/SSID
WlanConnectRespEvent	event informs if connect is in progress
WlanDisconnectReqEvent	event triggers disconnect to an AP/SSID
WlanDisonnectRespEvent	event informs if disconnect is in progress
WlanTrafficStatsEvent	event informs the traffic stats, tx and rx.
WlanResetEvent	event to trigger clean up all Wlan request

4.7.3. SYNC Applink

The NetworkInterfaceManager creates an NetworkInterface to track the state of Applink Virtual Network Interface. Applink in concert with Fordpass uses the Bluetooth or USB to provide access users cell phone data. Applink virtualizes this interface as an IP address to ECG-CM.

The interaction between the NetworkInterface instance and the Applink is mapped to the following events.

4.7.3.1. AppLink Events

Event	Description
AppLinkIfaceUpEvent	event informs the AppLink IP interface is Up
AppLinkIfaceDownEvent	event informs that AppLink IP interface is Down
AppLinkIfaceStateReqEvent	event triggers a request to get current AppLink Iface state
AppLinkIfaceStateRespEvent	event informs the state of AppLink Iface

4.7.4. VNM

The PathManager creates an instance Path to manage the creation of Tunnels and Tunnel end points and linking the tunnels and interfaces. The Path instance manages the creation of the tunnels and links using the VNM API's.

The interaction between the Path instance and the VNM is mapped to the following events.

4.7.4.1. VNM Events

Event	Description
VnmTunnelAddEpEvent	event triggers addition of tunnel endpoint on an ECU
VnmTunnelAddEpRespEvent	event informs if addition of an tunnel endpoint was a success or failure
VnmTunnelRemoveEpEvent	event triggers removal of tunnel endpoint on an ECU
VnmTunnelRemoveEpRespEvent	event informs if removal of an tunnel endpoint was a success or failure

VnmTunnelUpdateEvent	event triggers update of tunnel endpoint on an ECU
VnmTunnelUpdateRespEvent	event informs if update of an tunnel endpoint was a success or failure
VnmIfaceLinkEvent	event triggers linking of 2 interfaces (tunnel endpoints or a tunnel endpoint and an Iface)
VnmIfaceLinkRespEvent	event informs if linking of 2 interfaces was a success or failure
VnmIfaceLinkRemoveEvent	event triggers unlinking of 2 interfaces
VnmIfaceLinkRemoveRespEvent	event informs if unlinking of 2 interfaces was a success or failure

4.7.5. TCU AEC

The MainControl monitors the state of ECall, If ECall is in progress the MainControl and tears down all data calls that are active or in progress. All new data request are queued if ECall is in progress. When Ecall is over the all data calls and request are resumed.

The interaction between the MainControl and the AEC is mapped to the following events.

4.7.5.1. ECall Events

Event	Description
ECallIndicationEvent	event informs the current ECall state
ECallReqEvent	event triggers query of ECall state
ECallRespEvent	event informs the current state of the ECall

4.8. Power Manager

4.8.1. Ignition On

Regardless what is the previous power state of ECG-CM, when IGN_ON state ECG-CM provides connection services as ECG powered up. (Assumption: IGN_ON means all the systems are powered up)

ECG-CM listen to VPSM notification as an indication of ECG power state and further internal use.

4.8.2. Ignition Off and FPS to LPR

30 minutes Timer after IGN_OFF is not be handled by VPSM, so that ECG VPSM will notify CM when it is expire.

ECG-CM check if there is a existing connection, CM notify or request application to release connection. When all connections are released, CM notifies ECG VPSM or ECG PWR there is no connection exist as a part of voting to change power state.

4.8.3. Waking up TCU for SDN connection

When ECG-CM wakes up to provide application service to connect SDN, application should request CM for SDN connection. CM should request ECG VPSM or PWR module to wake up TCU to provide CM services. CM listens to TCU notification when it is ready.

"Application -> CM -> "VPSM or ECG-Pwr" -> TCU wakeup"

4.8.4. Voting after release connection:

CM will vote for Power State change. This voting rule is applied generally for CM because CM do not maintain previous power state of ECG or TCU. It measn that CM has no way to know previous Power states of other systems.

After application releases connections, CM notify ECG VPSM or ECG PWR there is no active connection. By doing this, CM vote to help power state change decision, but do not have a decision role whether power off TCU or put into LPR or any other power states.

> For example) ECG-CM wakeup -> FCI creates a connection -> FCI Release a connection -> CM notify (if there is no active connection) VPSM or PWR

Voting is generally applied when CM does not have an active connection regardless power state as well as after waking up from LPR.

4.8.5. Off Peak Ignition Off

ECG-CM is responsible for waking up the ECG for Off peak services during Ignition Off. It uses FTCP command to schedule an Off peak SMS wakeup from the SDN. The ECG-CM is responsible for managing the duration of Off Peak tasks. ECG-CM will interact ECG power management to control the power state of the ECG during Off peak Ignition Off.

4.9. FNV2-CM Entities

While ECG-CM is the chief component responsible for servicing requests for network interfaces from WIR Clients across all ECUs, the CM entities on non-ECG ECU's also need to take in account the use case where ECG might not be available or operational. In these scenarios, the FNV2-CM's will function in autonomous mode and would service requests and manage network interfaces locally. As a result, the FNV2-CM's wil be tasked with additional responsibilities.

4.10. ECall Handler

Refer to eCall(AEC) Detail Design

4.11. WLAN Profile Manager

CM WLAN Profile manager manages scannable external WLAN APs from TCU and possibly SYNC WLAN to provide wireless communication services. It maintains all the Aps' information including previously connected and newly scanned AP. It update profile list periodically or on demand as external AP environment is changing time to time. By doing this CM WLAN profile Manager maintains latest list and status of wireless characteristics such as RSSI or bandwidth information.

WLAN modules provide interface for related service requests/responses/events for profile state update. CM WLAN profile manager provides required information to WIR Clients. Profiles could be saved when it is known(once connected) profile into non-volatile memory for loading up after restart. There are multiple possible use cases;

4.11.1. Use cases

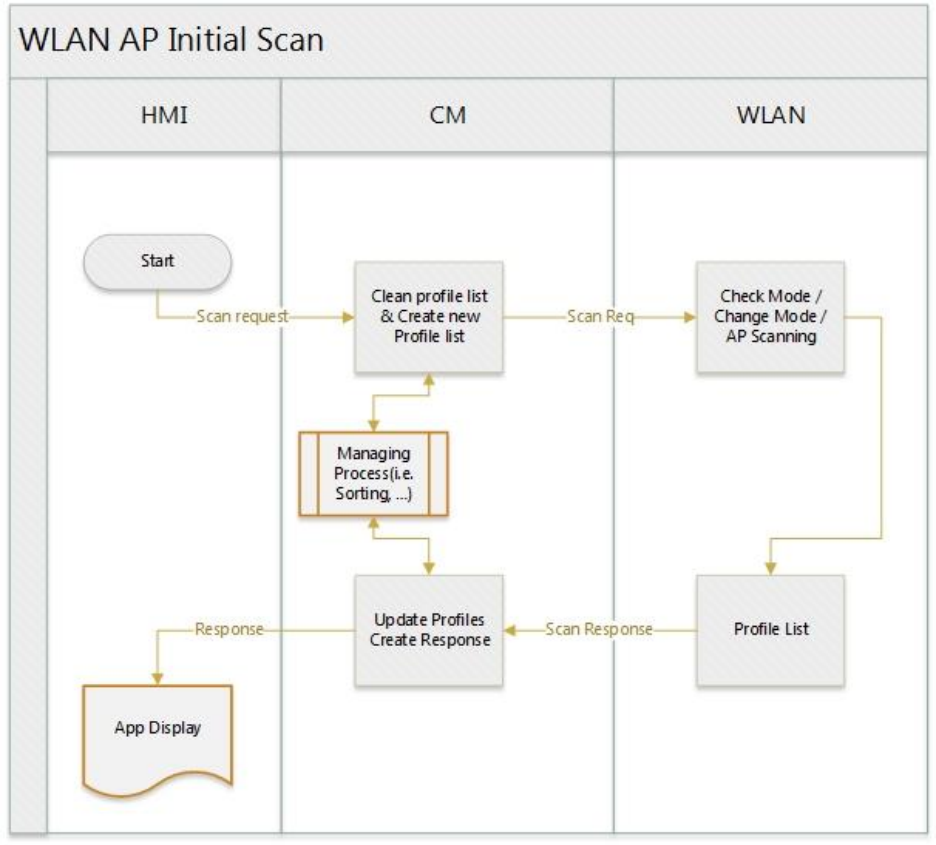

Figure 11: WLAN AP Initial Scan

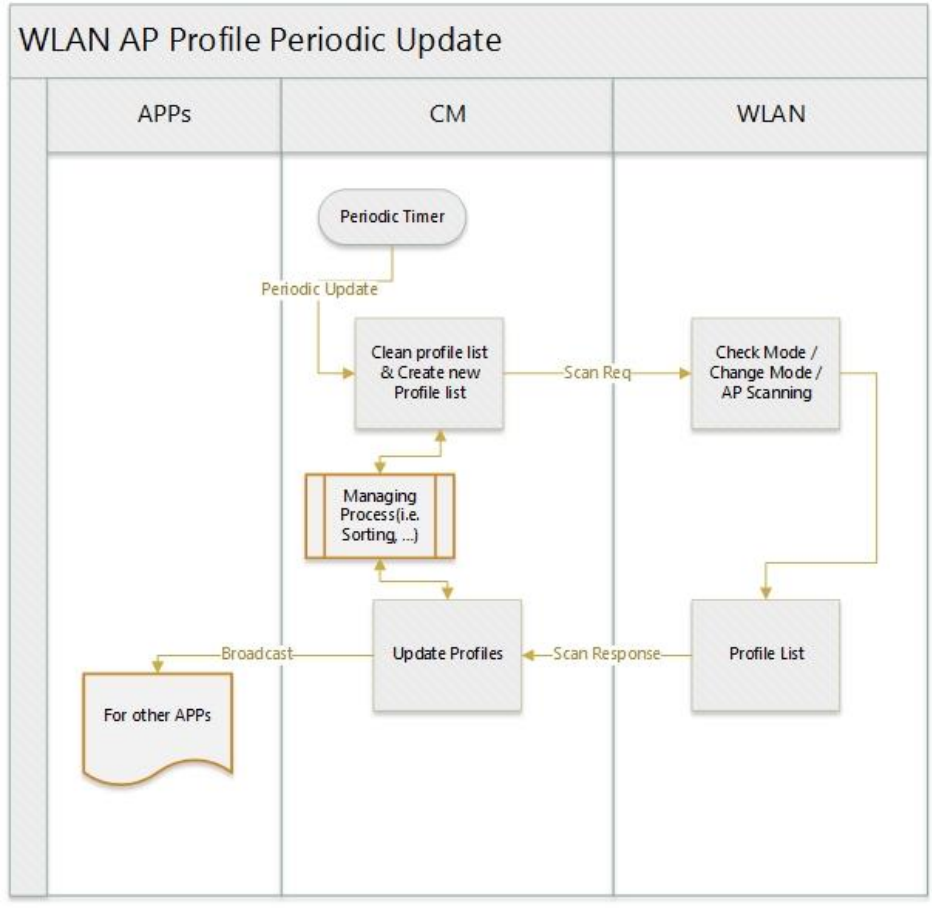

Figure 12: WLAN AP Profile Periodic Update

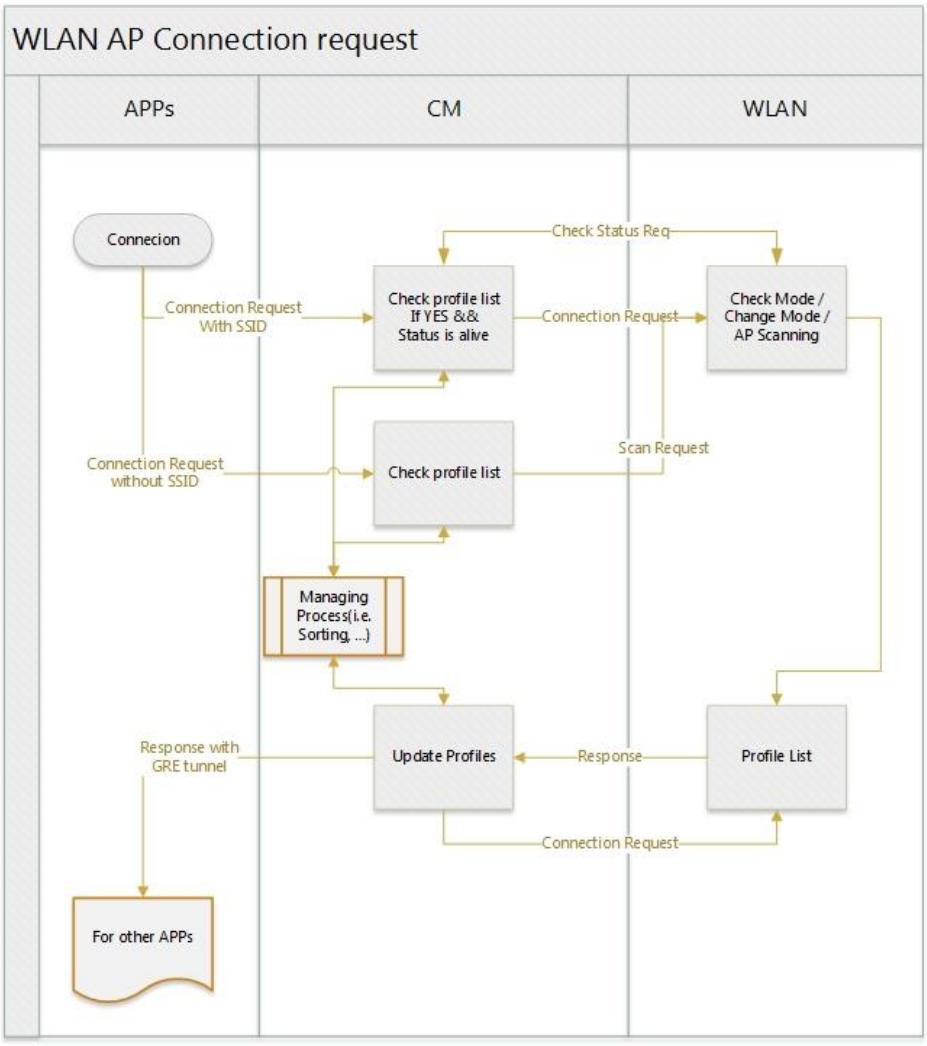

Figure 13: WLAN AP Connection Request

4.11.2. Data Class diagram

Profile lists are residing on both ECG-CM Master and clients on TCU, SYNC. Profiles in Master and Client should be synchronized at anytime when updated. The reason of maintaining profiles on both sides because Power Management requirement from WIR.

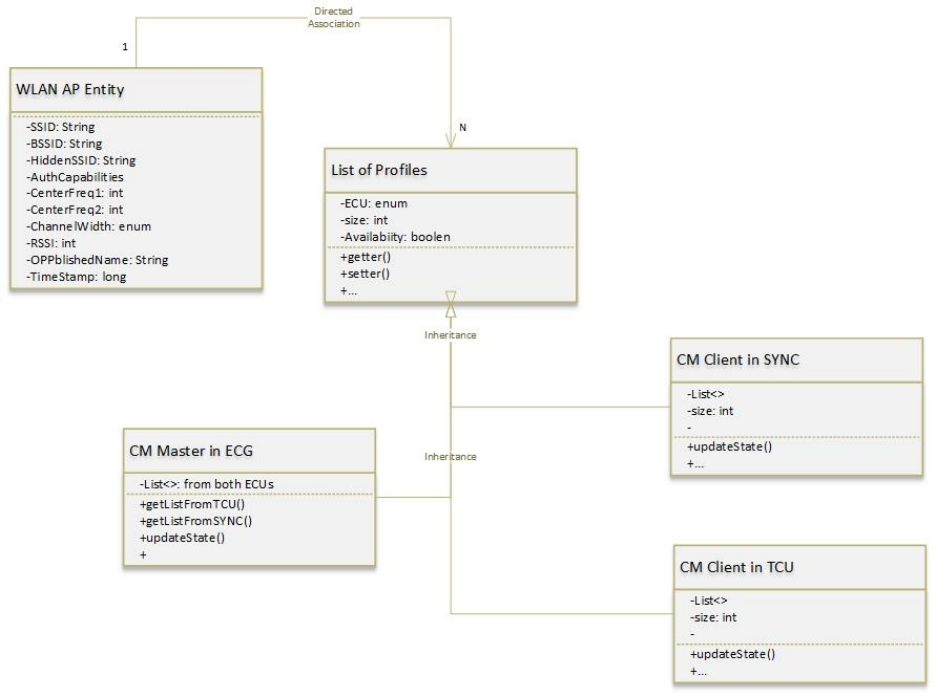

Figure 14: WLAN Profile Manager

4.11.3. WLAN Request and Response

4.11.3.1. Request / Response

sta_req_scan: Used by clients of WSTAS to request WSTAS to scan for near by networks

Details:

Topic	Type	Name
Topic	**Type**	**Name**
services/tcu/wlan/sta/request	request	sta_req_scan

Data:

Value	Description
Value	**Description**
<scan params>	Options used to define a scan criteria

Scan Params:

Parameter	Description
Parameter	**Description**
Passive	A flag indicating the scan is to be performed passively. By default, i.e. when omitting this parameter, scans are performed actively
Channel	A list containing 802.11 channels, spanning all bands, to scan
SSID	A list containing up to 10 network names to scan

sta_rsp_scan: Sent by WSTAS to the client that sent the sta_req_scan indicating that WSTAS has received the request and is starting to process it successfully

Details:

Topic	Type	Name
Topic	**Type**	**Name**
<chosen by sender of sta_req_scan>	response	sta_rsp_scan

Data:

Value	Description
Value	**Description**
success code (TBD)	WSTAS has received the scan request and has successfully started to process it.
failure code X	TBD
<integer>	Value to allow the original caller to identify which set of results is for their scan request.

4.11.3.2. WLAN message proto file

tcu-services/idl/wlan/**SoaWlanSta.proto**

5. Component Restart/Init/Bootup

5.1. ECG-CM

ECG-CM maintains a list of interfaces its managing, tunnels it brings up and any links established between tunnels and interfaces. ECG-CM also maintain a list of allocation requests from WIR Clients. ECG-CM does not persist any of the above information to flash. In the scenario where ECG-CM is stopped and restarted or crashes and restarts, it losses all the information relating to states of Interfaces, tunnels, links and allocation requests from WIR Clients. To handle this scenario ECG-CM will reset all the interfaces, tunnels, links and notify the WIR Clients on ECG-CM restart. This will bring all the components that ECG-CM interacts with to a known state.

ECG-CM will broadcast the (re)start information as an indication through SOA on a RESET topic. All components that interact with ECG-CM must subscribe this reset topic and perform clean up. For WIR Clients this means they must clean up any network sessions (close al open sockets) that are currently active. Clean up network connections requests to ECG-CM and reissue the allocation requests to ECG-CM. The reset behaviour will apply On first boot and after recovering a process crash or process restart.

SOA topic for reset broadcast : "SERVICES/DATA/ECG/CM/RESET"

5.1.1. VNM

For VNM this means that all tunnels endpoints need to be removed and all the links between tunnels and interfaces need to be unlinked. On receiving a reset from ECG-CM VNM will also send a SOA indication with the ECU's current IP address.

SOA topic for reset broadcast : "SERVICES/DATA/[ECU]/VNM/RESET"

5.1.2. DCM

ECG-CM reset, for TCU-Cellular DCM this means that all the cellular interfaces need to be cleaned up. After performing clean up DCM will send a reset indication with SESSION_INFO_IND with unique session ID. ECG-CM keep track of this session ID for its life cycle. Any time DCM sends SESSION_INFO_IND with a session ID, ECG-CM will compare it with current cached session ID. If cached session ID is different from the received session ID, ECG-CM will clean up all cellular interface references.

5.1.3. DCM-VNM

ECG-CM reset, since DCM is also responsible for VNM on TCU, this means that all tunnels endpoints need to be removed and all the links between tunnels and interfaces need to be unlinked.On receiving a reset from ECG-CM DCM-VNM will also send a SOA reset indication with the ECU's current IP address.

SOA topic for VNM reset broadcast: "SERVICES/DATA/TCU/VNM/RESET"

5.1.4. WLAN

For WLAN services this means that WLAN will dis-associate from the currently attached AP. WLAN on TCU will go back to the default mode WHS. ECG-CM will re request STA mode for WLAN on TCU and association based on WIR Clients request.

5.1.5. AEC

AEC, no impact on AEC. ECG-CM on first boot (or any restart) will query the state of AEC before handling any apllication data requests.

5.1.6. VPSM

ECG-VPSM, ECG-CM will query the state of ECG-VPSM to determine ECG power state.

5.1.7. Applink

Applink interaction is TBD, Applink design in progress. Since ECG-CM does not control the state of Applink, ECG-CM will most likely require a querying the state of Applink virtual interface.

5.1.8. ECG-CM Client

TBD

5.2. Other Components

ECG-CM will needs to keep track of VNM, TCU DCM and WLAN services on TCU and SYNC, Applink and all WIR Clients that request connections. ECG-CM expects that the software components it is interacting with will notify ECG-CM whenever the component restarts (first boot, crash or controlled restart). The current mechanism in place is to listen to a SOA reset indication from these components. ECG-CM will subscribe to these SOA reset notifications and cleanup ECG-CM references related to that component.

5.2.1. VNM Restart

VNM reset is ECU specific, ECG-CM on receiving reset from VNM will be notify the WIR Clients that the interface is down. ECG-CM will recreate any tunnel endpoints that were previously present on that VNM. It will also setup any links that are were present between tunnel endpoints and an interface (or another tunnel endpoint). . ECG-CM will re request the setup of tunnels and links based on the current WIR Client requests. Once the tunnels are establised the WIR Clients will be notified that the interface is up.

5.2.2. DCM Restart

On DCM restart DCM will send SESSION_INFO_IND with unique session ID. Any time DCM sends SESSION_INFO_IND with a session ID, ECG-CM will compare it with current cached

session ID. If cached session ID is different from the received session ID, ECG-CM will clean up all cellular interface references. ECG-CM will cleanup all cellular interface state machines. WIR Clients are notified that the interface is down. ECG-CM will re request the cellular interface bring up. once the cellular interface is up and tunnels created and linked, ECG-CM will notify the apps that interface is up.

5.2.3. DCM-VNM Restart

DCM-VNM will send a SOA reset indication with the ECU's current IP address. ECG-CM on receiving reset from DCM-VNM will be notify the WIR Clients that the interface is down. ECG-CM will recreate any tunnel endpoints that were previously present on that DCM-VNM. It will also setup any links that were present between tunnel endpoints and an interface (or another tunnel endpoint). ECG-CM will re request the setup of tunnels and links based on the current WIR Client requests. Once the tunnels are established, the WIR Clients are notified that the interface is up.

SOA topic for VNM reset broadcast: "SERVICES/DATA/TCU/VNM/RESET"

5.2.4. WLAN Restart

WLAN Services reset is ECU specific,ECG-CM will cleanup the ECU specific WLAN state machines and notify the WIR Clients that interface is down. ECG-CM will re request WLAN services bring up and

5.2.5. AEC Restart

AEC reset. ECG-CM will listen to AEC SOA topics for a restart indication, ECG-CM will clean up the its current AEC state to allow WIR Clients data requests.

5.2.6. VPSM Restart

ECG-VPSM rest , ECG-CM will clen up the current power state and listen to ECG-VPSM for new state transitions.

5.2.7. ECG-CM Client Restart

TBD

6. Reference

6.1.1. WiFi STA(Station or Client) feature design document

WLAN Service HLD:
https://www.eesewiki.ford.com/display/tcu/WLAN+Service+in+the+TCU#WLANServiceintheTCU-Provisioning.1

WHS intf HLD:
https://www.eesewiki.ford.com/display/tcu/WHS+Intf.+in+the+TCU#WHSIntf.intheTCU-Provisioning/EOL

STA intf HLD:
https://www.eesewiki.ford.com/display/tcu/STA+Intf+in+the+TCU#STAIntfintheTCU-Provisioning/EOL