# BeeRocks V1.4 Architecture Review History

| Rev | Description | Polarion Version | Date |
|---|---|---|---|
| 0.1 | First draft | | 01-Jul-2018 |
| 0.2 | Update GW / IRE boot and onbording flows | 170250 | 16-Jul-2018 |
| 0.3 | Update controller discovery and AP-Configuration | 171542 | 2-Aug-2018 |
| 0.4 | Added M1/M2 WSC processing flows | 202940 | 19-Nov-2018 |
| 0.5 | Add new operating modes | | 20-Jan2019 |

# Table of Contents

## Contents



## 1 Introduction

This document provides a functional description of BeeRocks v1.4 system
including: architecture, features, WFA EasyMesh / Multi-AP v1.0 standard support and system end to end flows.

### 1.1 Scope and Purpose

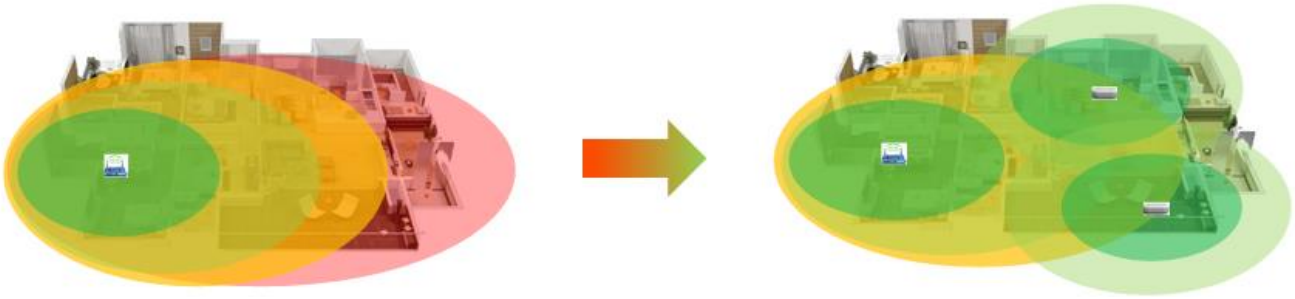
This document is focused on describing the high level architecture and flows for BeeRocks core that implements WFA Multi-AP (EasyMesh) controller & agent including Intel enhancements. The document allows engineers to understand: high level architecture, brake down to modules, modules communication, supported features and system flows.

### 1.2 Introduction to WFA EasyMesh / Multi-AP

As part of Intel's effort to socialize/standardize operation of simultaneous multiple AP in the home Intel created a spatial interest group (SIG) that focused on this topic. The SIG work was eventually contributed to WFA creating the EasyMesh R1.0 standard.
The main goal of the standard is to:

1. In order to ensure whole home coverage customers and service providers are increasingly using multiple access points in a home.
2. While multiple access points will increase coverage, without coordination of the access points the full benefits of increased throughput may not be realised.

3. To date no standard exists to allow access points from different vendors to on-board, discover, configure and communicate with each other in order to optimize the in-home Wi-Fi experience



### 1.2.1 Multi-AP Logical Entities

Multi-AP network consists of two types of logical entities:

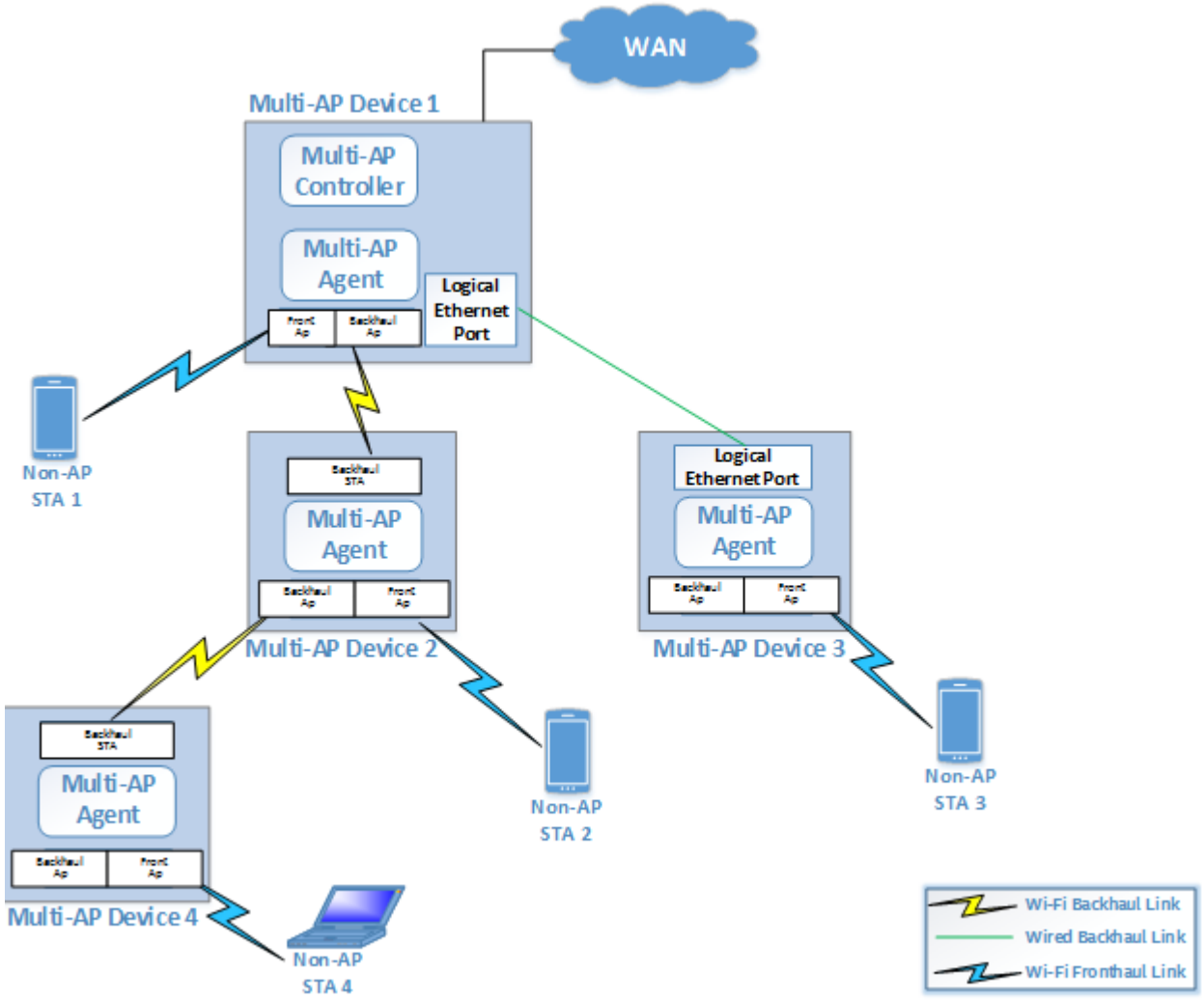
One Multi-AP Controller:
Multi-AP Controller is a logical entity in a Multi-AP network that implements logic for controlling the fronthaul APs and backhaul links in the Multi-AP network.
A single Multi-AP Controller is supported for a given Multi-AP network. A Multi-AP Controller receives measurements and capability data for fronthaul APs, clients and backhaul links from the Multi-AP Agents and triggers AP control related commands and operations on the Multi-AP Agents.
Multi-AP Controller also provides onboarding functionality to onboard and provision Multi-AP devices onto the Multi-AP network.

One or more Multi-AP Agents:
A Multi-AP Agent is a logical entity that executes the Multi-AP Controller functions, and reports measurements and capabilities data for fronthaul APs, clients and backhaul links to a Multi-AP Controller and/or to other Multi-AP Agents.
A Multi-AP Agent interfaces with Wi-Fi sub-systems for fronthaul APs and backhaul STA on the Multi-AP device to get measurements and capabilities data, apply configuration changes and execute AP control functions.

### 1.2.2 Multi-AP – Network Device

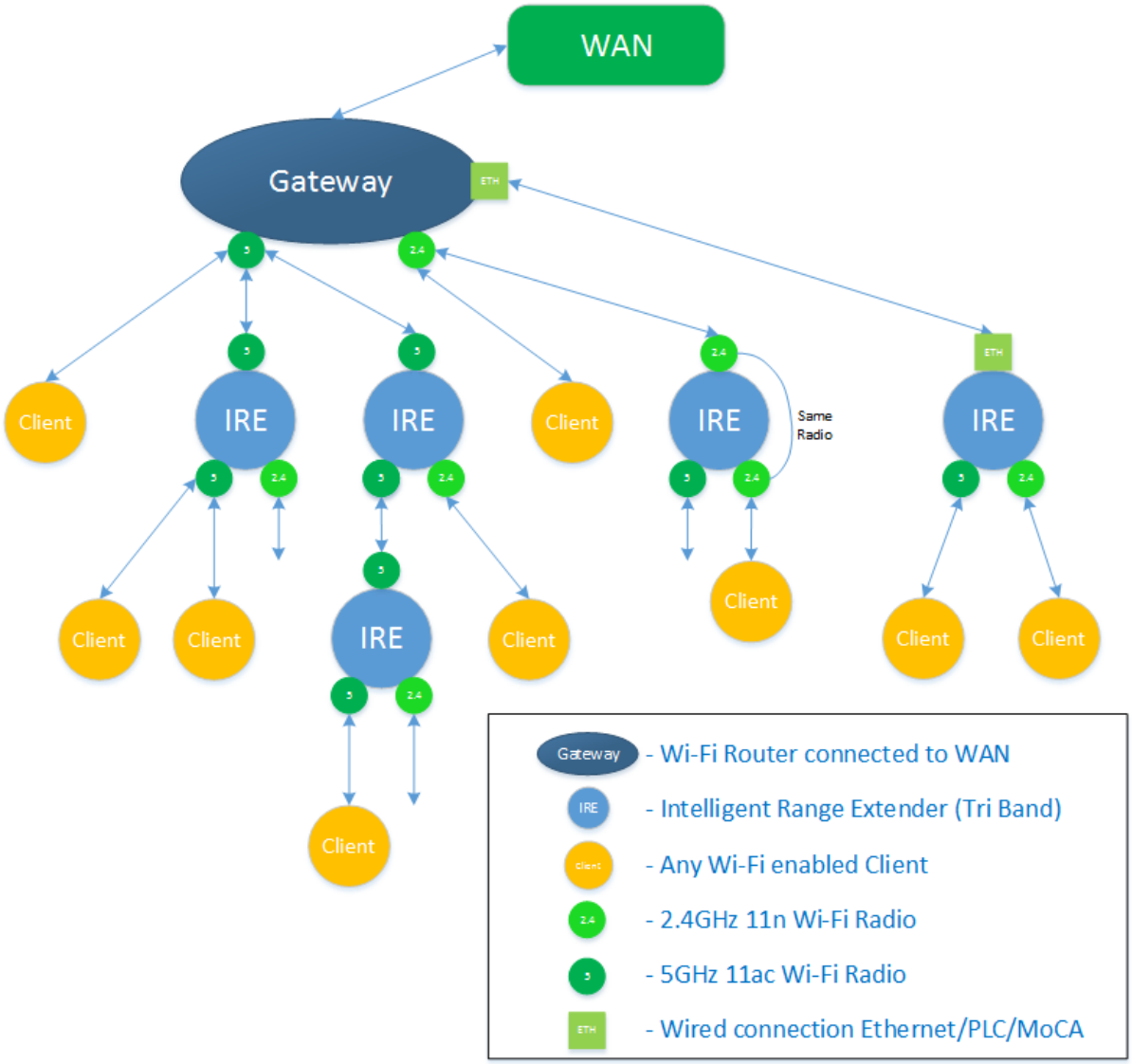
- A Multi-AP device may contain a Multi-AP Controller only, a Multi-AP Agent only or both Multi-AP Controller and Multi-AP Agent.
- Two Multi-AP devices with Multi-AP Agents connect to each other over a backhaul link which could be either a WiFi link or a wired logical Ethernet link. A single active backhaul link is allowed between any two Multi-AP devices at any given time.
- A Multi-AP device with a Multi-AP Agent includes fronthaul AP(s) for client STAs and/or a backhaul STA to associate with for Wi-Fi backhaul connectivity. In cases where a Wi-Fi backhaul link is supported, a Multi-AP device with a Multi-AP Agent also includes a backhaul STA to enable Wi-Fi backhaul link with another upstream fronthaul AP.
- Multi-AP devices with Multi-AP Agent functionality are connected to each other in a tree topology over one or more hops within a Multi-AP network. The tree topology ensures that a single backhaul path (over one or more hops) is established between any two Multi-AP devices in a Multi-AP network.
- Multi-AP selected IEEE 1905.1 as it's L2 communication protocol. Allowing reuse and extension of the existing 1905.1 stack.

Multi-AP – Typical Deployment:




## 1.3 Network Topology

Once GW is up IRE‘s and clients start to connecting to the GW, IRE’s are directly managed by the BeeRocks master controller setting their backhaul connectivity endpoint (Ethernet or Wi-Fi) and their front radio operating channel. For every IRE that joins the network master initiate optimal path selection that steers the IRE to the optimal location in the network.This is referred to as SON - Self Organizing Network, there are other factors that are contributing to the steering decision like new channel selection and load balancing. Tree topology in BeeRocks supports up to 4 IRE’s, configured in a hierarchy of up to 2 backhaul hops (in future releases this limitation might be expanded). As it will be described later, the tree topology is not fixed, but can be dynamically reconfigured in accordance with the strength of the bachaul links at any given time.



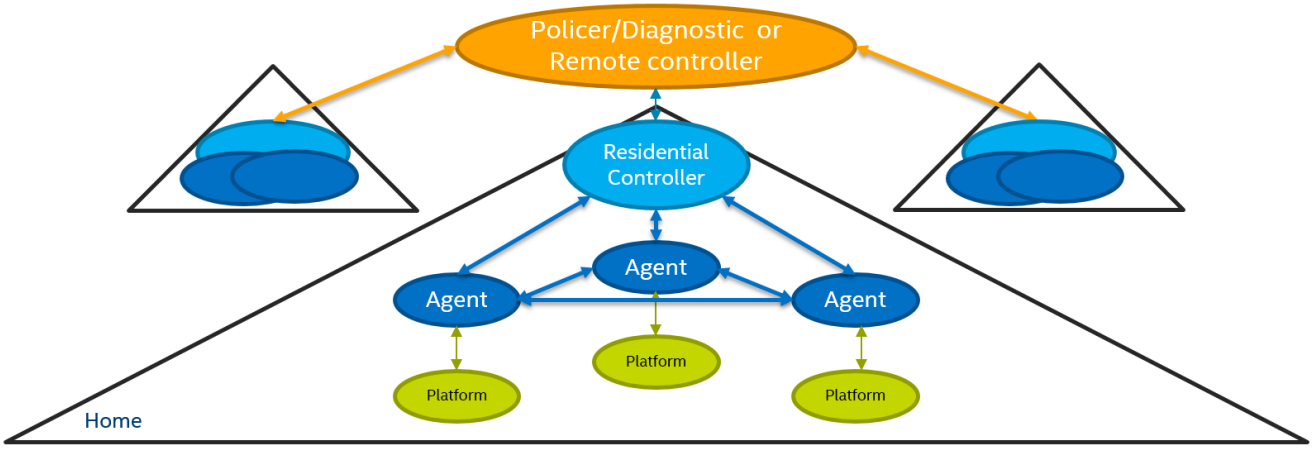

## 1.4 System Architecture Concept

BeeRocks v1.4 architecture must facilitate Multi-AP support allowing platforms running BeeRocks v1.4 to pass WFA certification while maintain BeeRocks v1.4 features such as legacy steering and DFS channel selection. While platform must support BeeRocks v1.4 agent / controller architecture it must also support 3rd party solution to replace controller or agent/controller .

Architecture goals:

- Ease integration of 3rd party solutions for Multi-AP and proprietary stacks.
- Multi-AP v1.0 Certifiable agent and controller
- Multi-AP v1.0 Interoperability (+enhancments)
- Extendable services on top of Multi-AP (customizable)
- Extend WRT services to all IRE, dWRT (customizable)
- Enable Multi-AP transport and alternative transport (customizable)

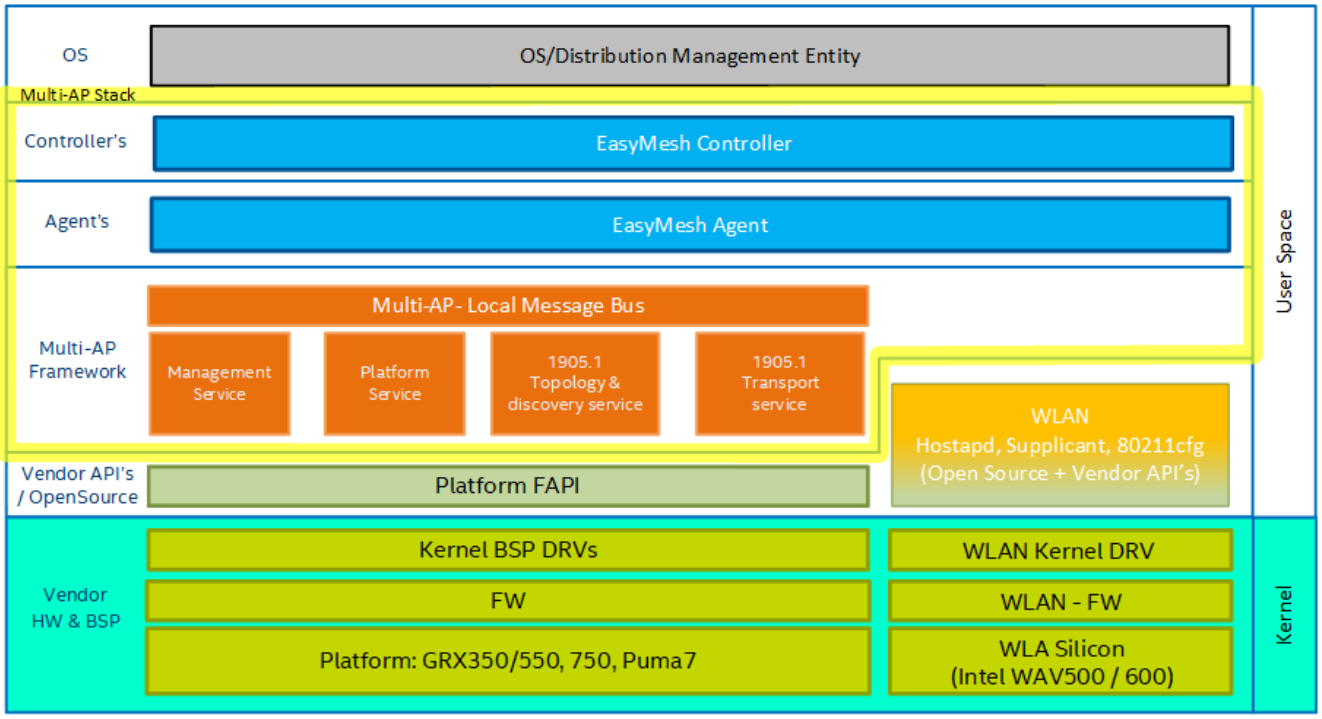
High level architecture partitions the entities in to 4 layers as follows:




Agent: is abstraction to platform to remote interface, south APIs (WFA Multi-AP)
Controller: is the decision making management entity (WFA Multi-AP)
Policer: is cloud remote manage entity (not defined)

## 1.5 System Architecture Partitioning

System Architecture is designed to allow plug-able services design where service components can be swapped at every level allowing flexible integration options for the OEM. Following diagram shows the architecture solution in layers where each layer offers additional functionality.




## 1.6 Supported Hardware

Multi-AP framework and core (BeeRocks V1.4) are designed to support any Linux based distribution and any WLAN solution. Specifically the main development platform is Easy-350 and Wi-Fi chip set is WAV500 2.4G &5G.
Supported Intel platforms will induce: GRX-350/550/750, Puma7, FLMX and WAV500 / WAV600 Wi-Fi.

# 2 Features and Requirements

## 2.1 WFA EasyMesh High Level Feature

- On-boarding: A new ‘multi-AP’ device entering the home gains layer 2 connectivity.
- Discovery: A new ‘multi-AP’ device establishes its role as ‘Controller’ or ‘Agent (Controlee)’.
- Configuration: A new ‘multi-AP’ device is provided with the configuration for the home e.g. SSID naming.
- Channel selection: coordinated channel selection to minimize co-channel interference.
- Capability reporting: understanding the capabilities of all the other access points in the ecosystem.
- Link metric reporting: quantifying the link capability between access points.
- Client steering: to position clients on the most advantageous access point/band.
- Backhaul optimization: providing robust inter access point links.
- Higher layer data payload: to provide extensible.

## 2.2 Intel Mesh Features

BeeRocks features are described in details in the IRE MRD document: IRE Features Spec

The only new feature in this version is changing the backhaul connection from a shared front/back VAP connection to a separate backhaul network for IRE connection. More details are available in the System flow sections.

## 2.3 Requirements

### 2.3.1 Multi-AP Framework

Multi-AP framework requirements are defined in EasyMesh Framework Requirements document.

### 2.3.2 BeeRocks Requirements

BeeRocks requirements are defined in BeeRocks_Requirements document, all new items are marked with V1.4.

BML V1.4 API's are defined in BML_API_Reference

BPL V1.4 API's are defined in BPL_API_Reference

### 2.3.3 WLAN Subsystem Requirements

WLAN subsystem requirements are defined in WLAN_Requirements document, all new items are marked with V1.4.

### 2.3.4 Platform Requirements

Platform requirements are defined in BeeRocks_Requirements document, all new items are marked with V1.4

# 3 Multi-AP Framework Architecture

Multi-AP framework architecture is documented here EasyMesh Framework Architecture

# 4 BeeRocks Architecture

## 4.1 Overview

BeeRocks is designed to run on any Linux based networking device. Design takes in to considerations needed platform / HW abstraction so that platform porting effort is constrained to minimum number of defected components. As described in the high level architecture intro, BeeRocks is divided to controller and agent components where agent runs on all network nodes and controller runs only on one device GW or IRE (AKA IRE master mode).

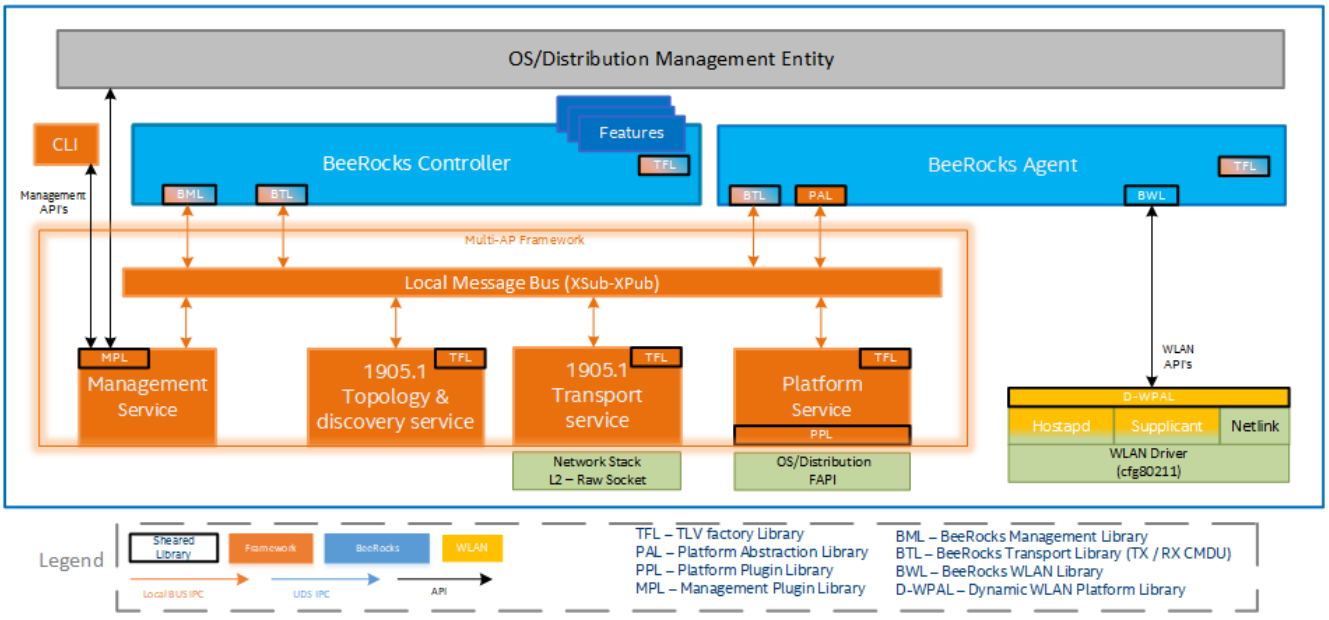
BeeRocks internal IPC is done via Unix Domain Sockets (UDS) which are very portable and efficient. BeeRocks has a communication thread base class that UDS communication with reusable wrapper class.

External communication to the EasyMesh framework is done via the framework local BUS.

## 4.2 Multi-AP Deployment Modes

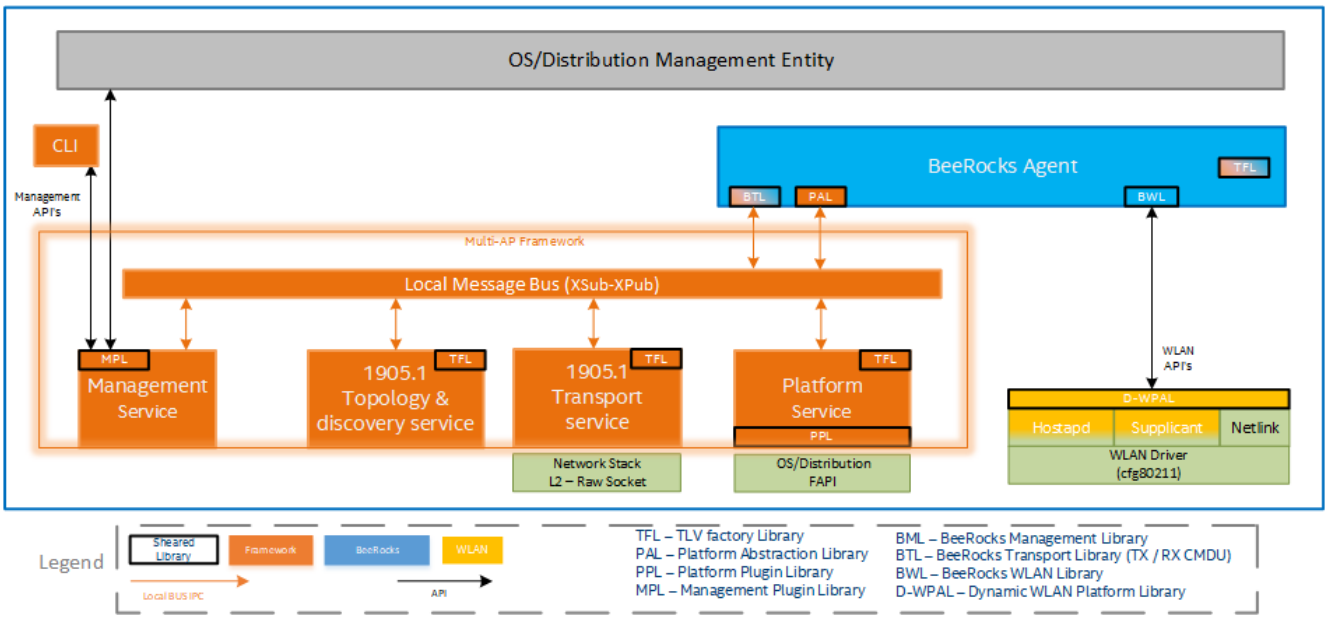
To allow a flexible deployment of the Multi-AP stack that meets the needs of different customers the stack defines four deployments mode:

**EasyMesh Managed Mode:**
This is the full Multi-AP mode there the framework with 1905.1 + BeeRocks agent + controller is deployed. Communication between components is made via the XSub/XPub local bus.



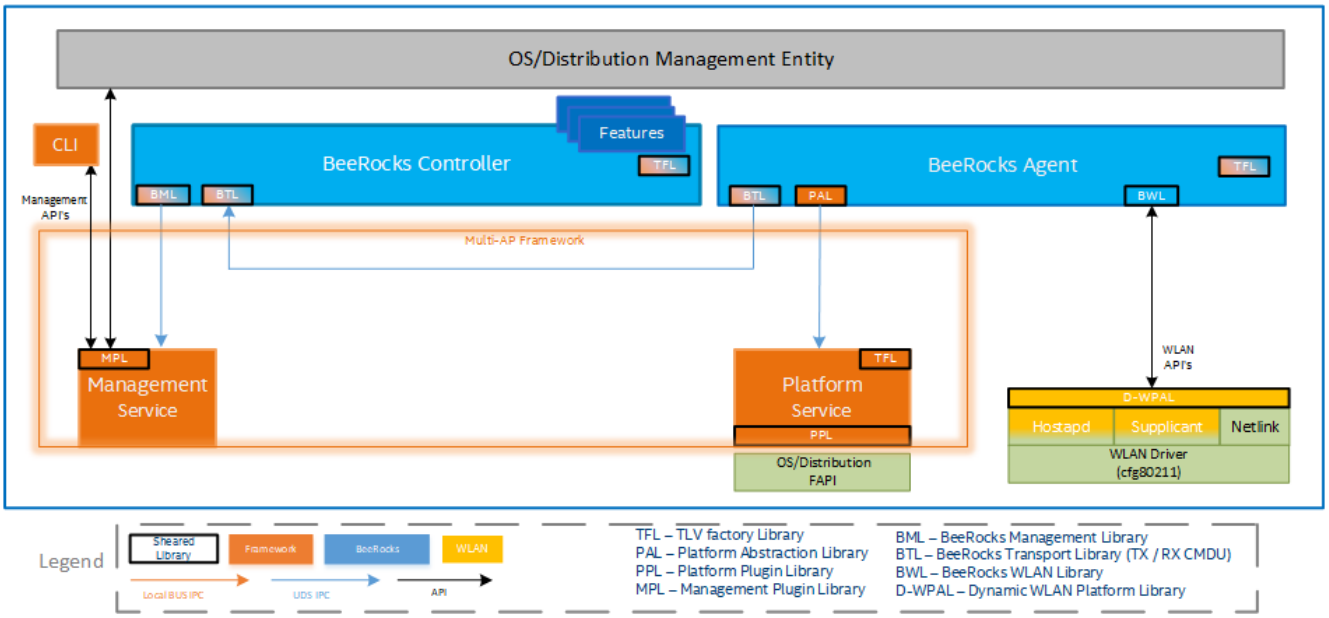

**EasyMesh Unmanaged Mode:**

This mode is identical to the EasyMesh managed mode except the controller is not deployed. It is assumed that an external controller is attached to the local bus or compliantly external to the stack.



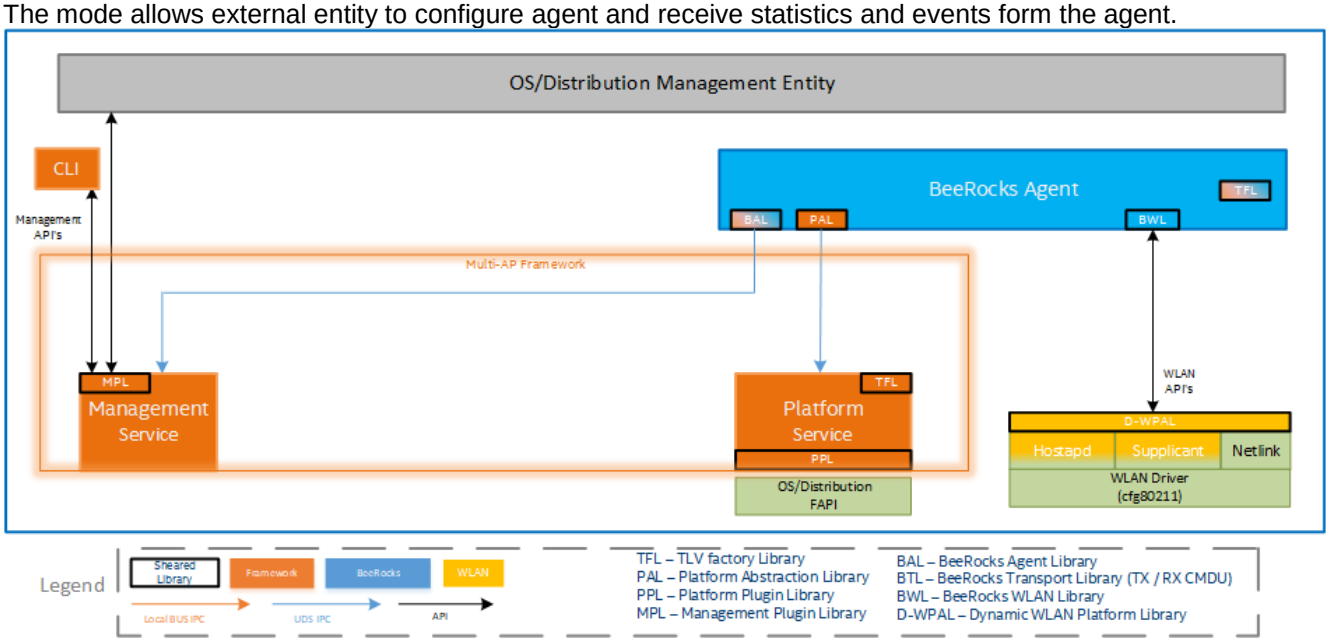

**None Mesh Managed Mode:**

This mode allows non mesh deployment where 1905.1 components are not deployed.
BeeRocks controller manages the GW local radios according to enabled features. There may be a case where there is an upper layer controller operating alongside the BeeRocks controller, in this case the enabled features of each must not crate a contention.



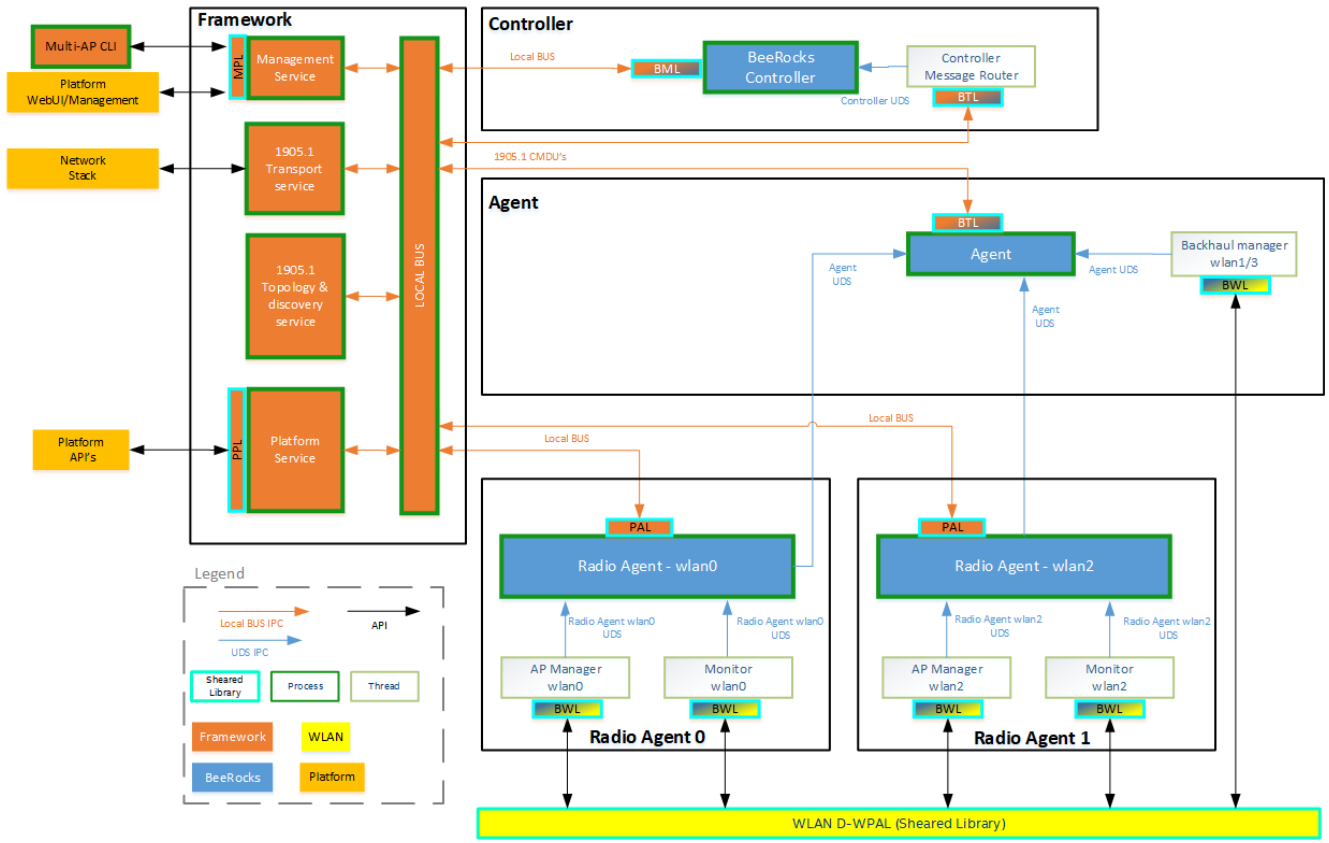

**None Mesh Unmanaged Mode:**

This mode is identical to the None Mesh Managed mode except the controller is not deployed.
The mode allows external entity to configure agent and receive statistics and events form the agent.

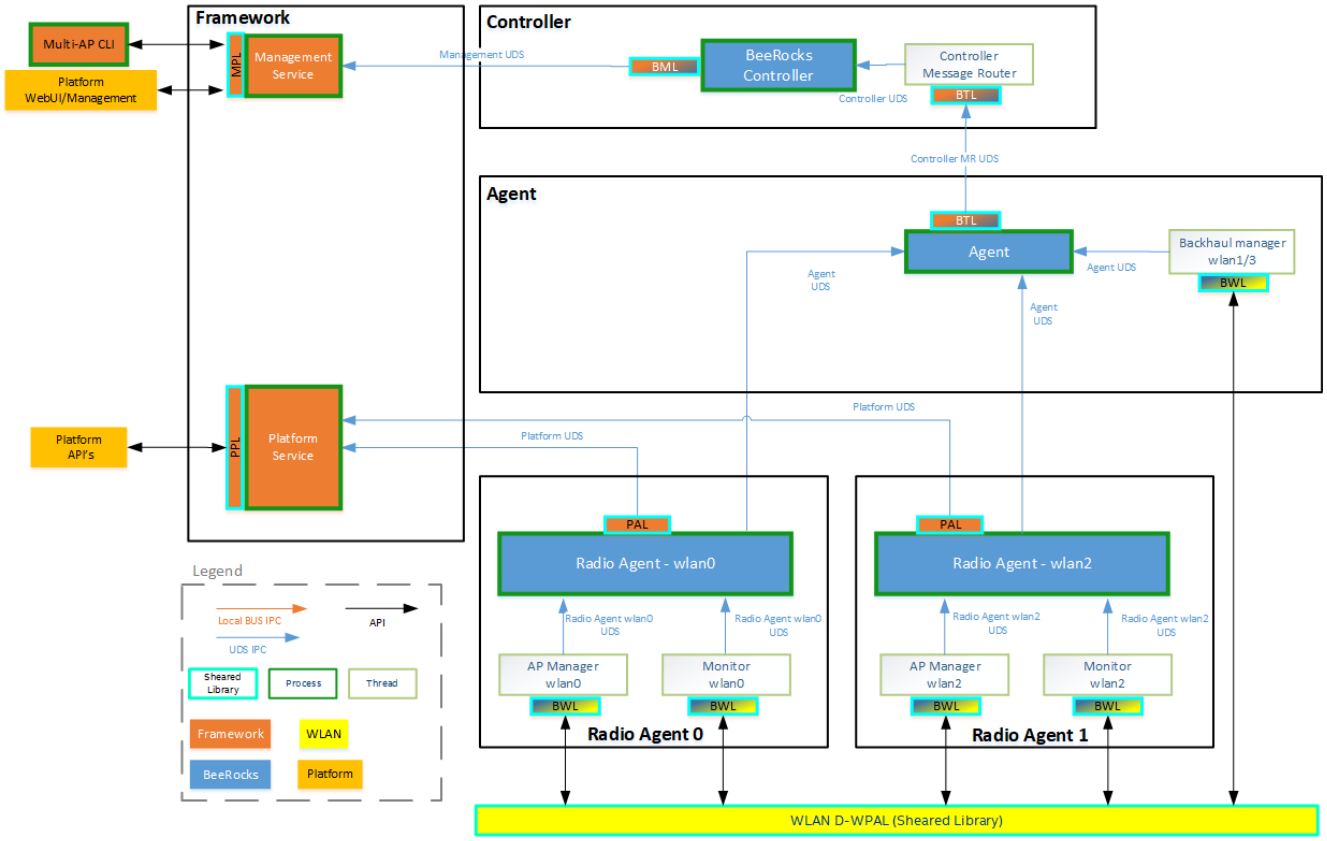



## 4.3 BeeRocks Inter/Outer Communication (IPC)

The following figure shows all inner module IPC communication of the Multi-AP stack and BeeRocks when the EasyMesh mode is deployed:

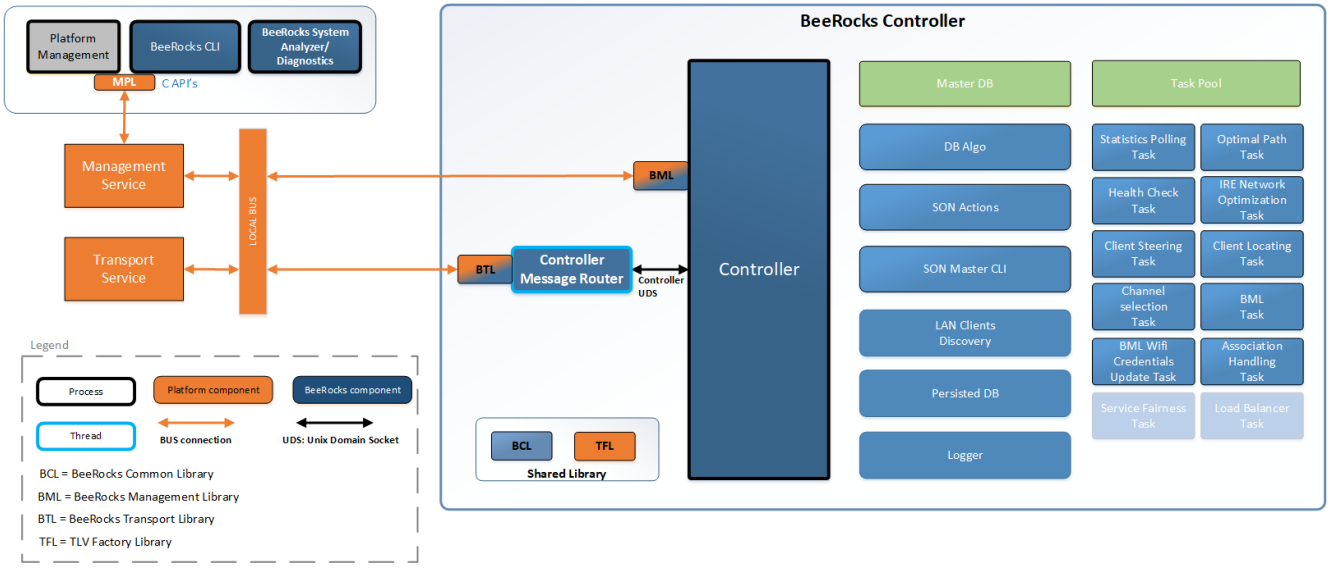



For the Non Mesh mode where the local bus and 1905.1 are not deployed the communication is done using p2p UDS carring the came vendor specific CMDU's:

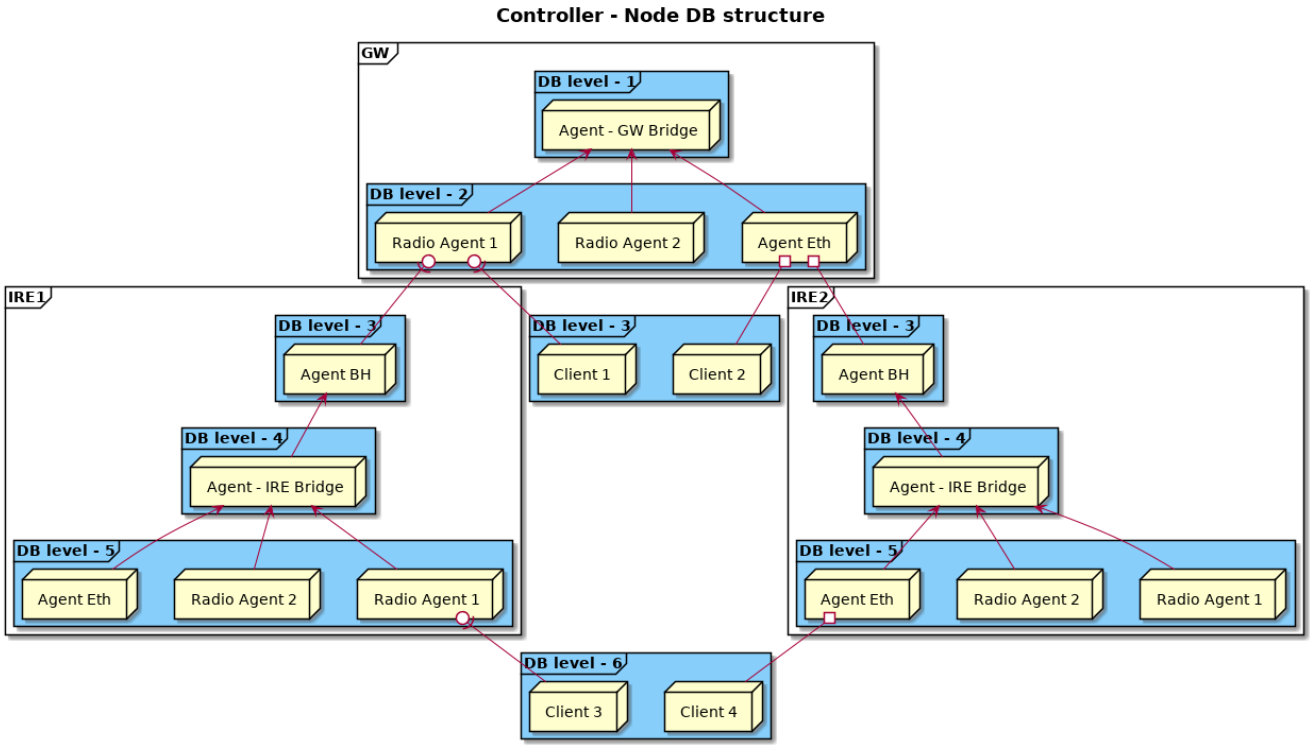



## 4.4 BeeRocks Controller

~~BeeRocks v1.4 controller will maintain its current architecture in respect to modules / task structure and will introduce a new transport library (BTL) that will allow the controller to communicate with the local 1905.1 transport agent via the local message bus. This library will be integrated to existing controller message router module. The transport library is used by controller and agent and allow abstraction from the platform specific implementation.~~



### 4.4.1 Controller Database Structure

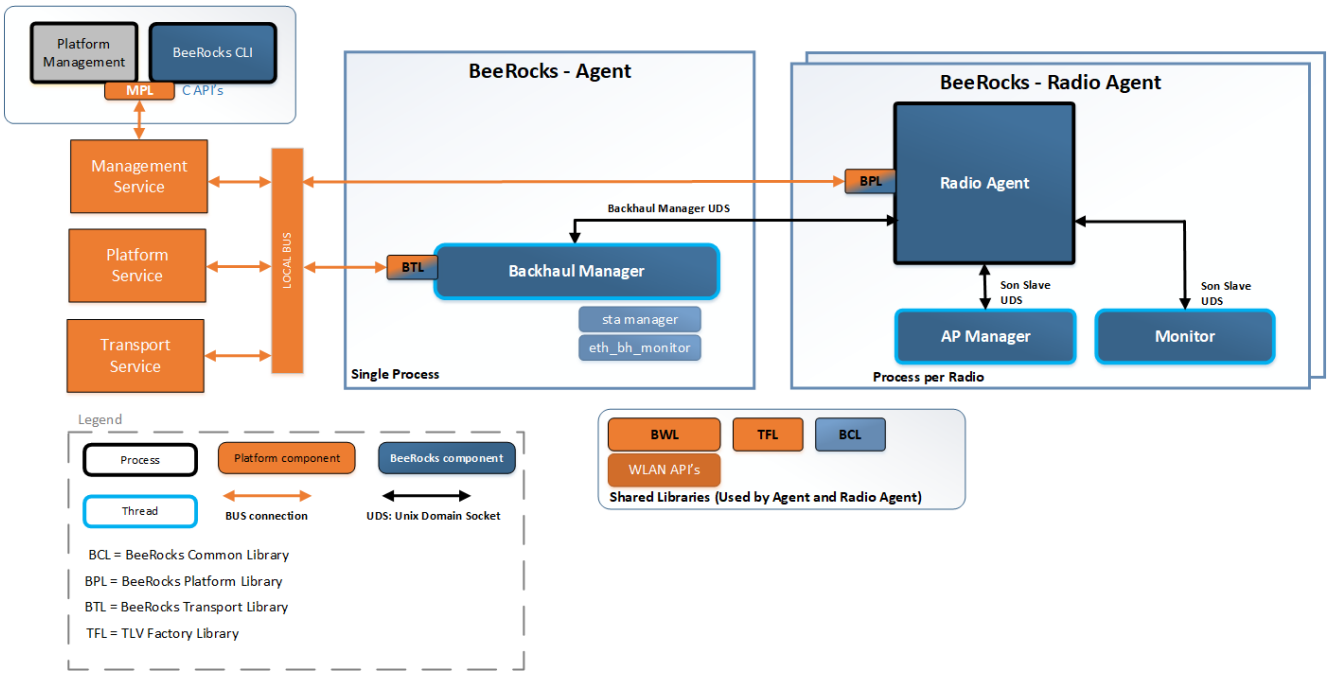



### 4.4.2 Controller UDS Socket Management

## 4.5 BeeRocks Agent

~~Similar to BeeRocks controller the agent will integrate the new BeeRocks Transport Library (BTL) that will allow communication with platform 1905.1 or alternative transport service.~~

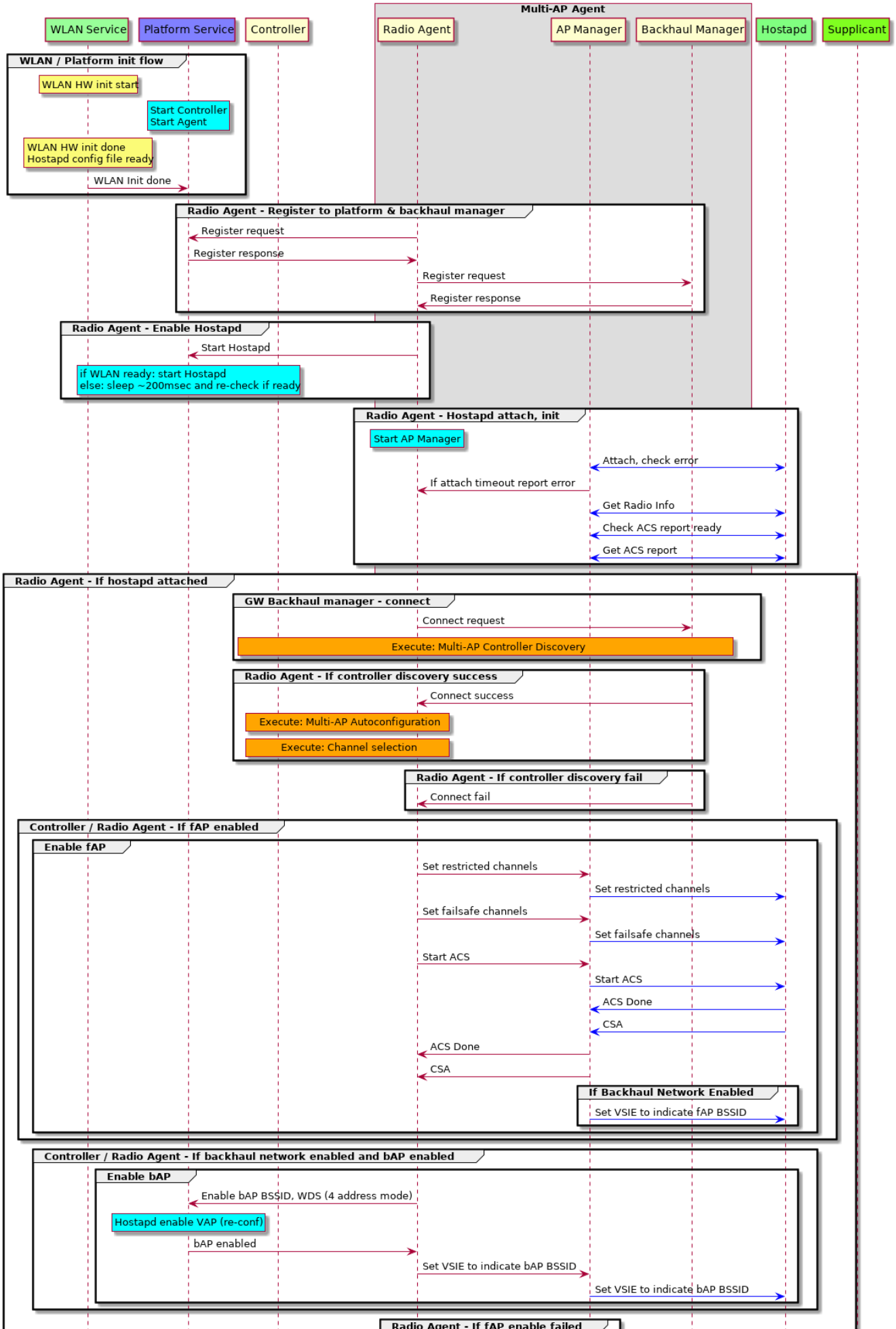
~~All agent specific API's will be moved to new library named BeeRocks Agent Library (BAL). This will allow the agent to be independent from controller, aligning to the requirement to have separate packaging for controller and agent.~~




# 5 BeeRocks Flows

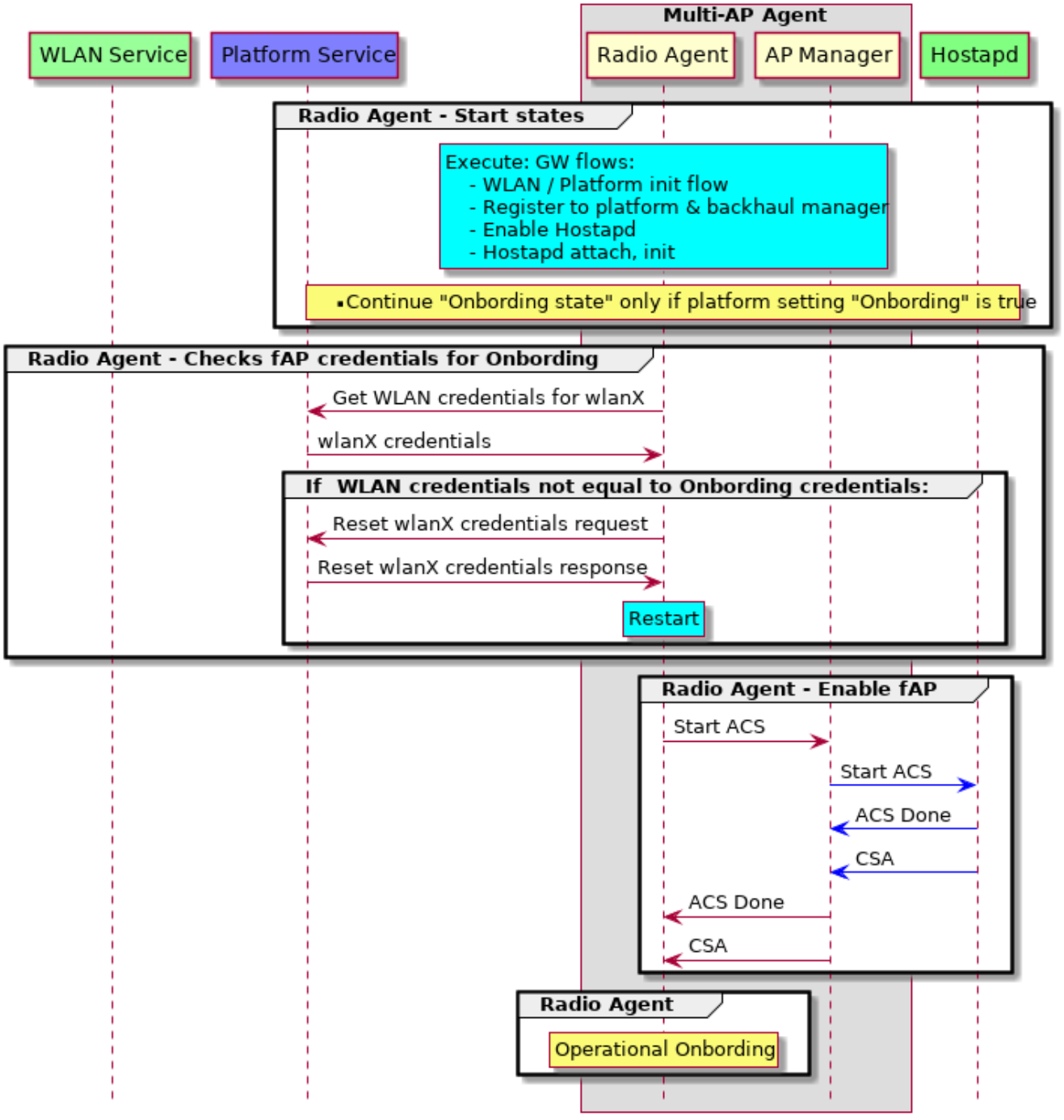
## 5.1 GW Boot

The following high level flow diagram describes the GW boot flow inducing interaction between the different entities in the system.


Multi-AP Agent
WLAN Service
Platform Service
Controller
Radio Agent
AP Manager
Backhaul Manager
Hostapd
Supplicant
WLAN / Platform init flow
WLAN HW init start
Start Controller
Start Agent
WLAN HW init done
Hostapd config file ready
WLAN Init done
Radio Agent - Register to platform & backhaul manager
Register request
Register response
Register request
Register response
Radio Agent - Enable Hostapd
Start Hostapd
if WLAN ready: start Hostapd
else: sleep ~200msec and re-check if ready
Radio Agent - Hostapd attach, init
Start AP Manager
Attach, check error
If attach timeout report error
Get Radio Info
Check ACS report ready
Get ACS report
Radio Agent - If hostapd attached
GW Backhaul manager - connect
Connect request
Execute: Multi-AP Controller Discovery
Radio Agent - If controller discovery success
Connect success
Execute: Multi-AP Autoconfiguration
Execute: Channel selection
Radio Agent - If controller discovery fail
Connect fail
Controller / Radio Agent - If fAP enabled
Enable fAP
Set restricted channels
Set restricted channels
Set failsafe channels
Set failsafe channels
Start ACS
Start ACS
ACS Done
CSA
ACS Done
CSA
If Backhaul Network Enabled
Set VSIE to indicate fAP BSSID
Controller / Radio Agent - If backhaul network enabled and bAP enabled
Enable bAP
Enable bAP BSSID, WDS (4 address mode)
Hostapd enable VAP (re-conf)
bAP enabled
Set VSIE to indicate bAP BSSID
Set VSIE to indicate bAP BSSID
Radio Agent - If fAP enable failed

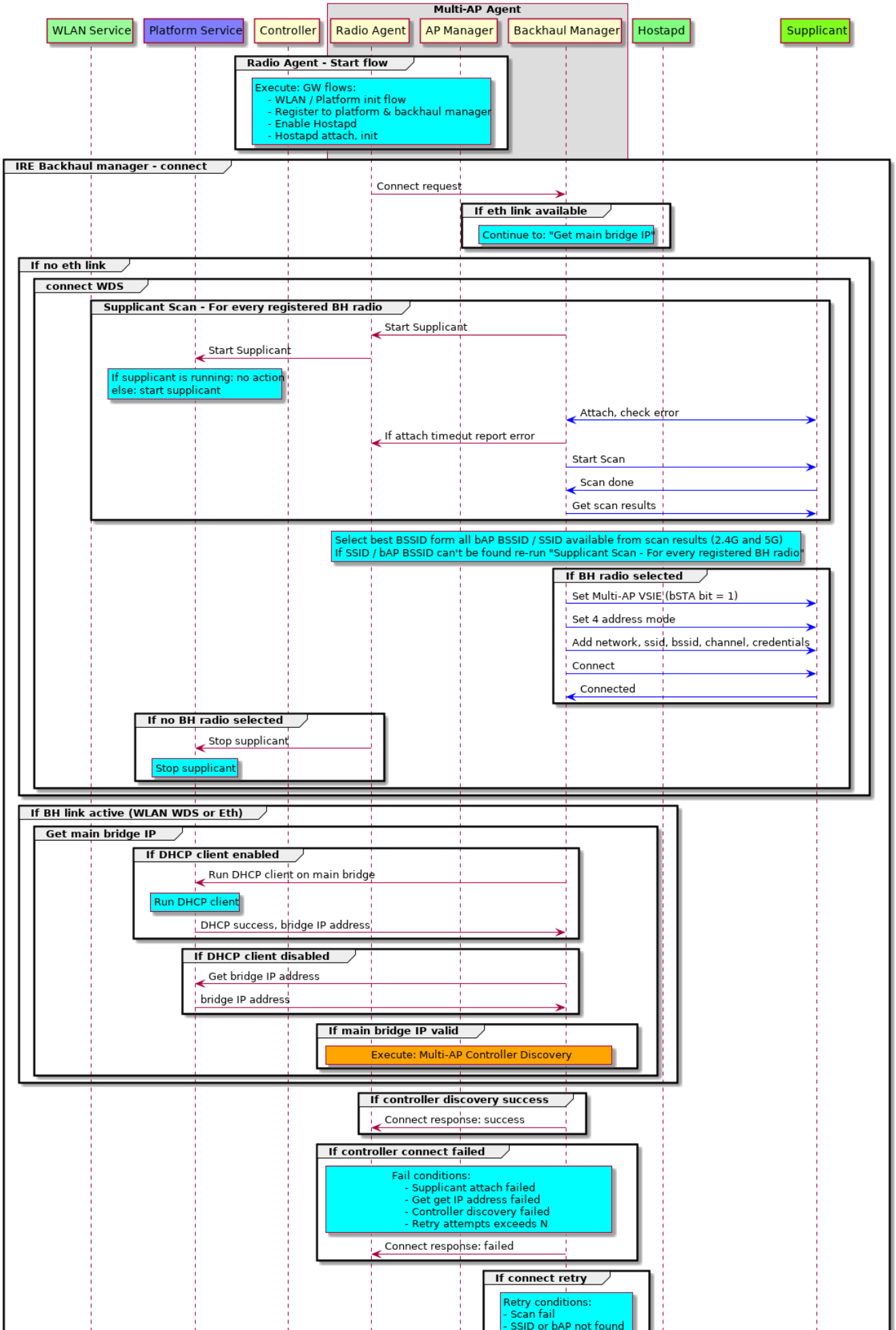
## 5.2 IRE Boot - Onbording state

The following diagram describes the flow of the agent when "onbording" state is set to "true" i.e. prior to controller auto-configuration.




## 5.3 IRE Boot - Operational

The following high level flow diagram describes the IRE boot flow using WLAN or Ethernet backhaul inducing interaction between the different entities in the system.

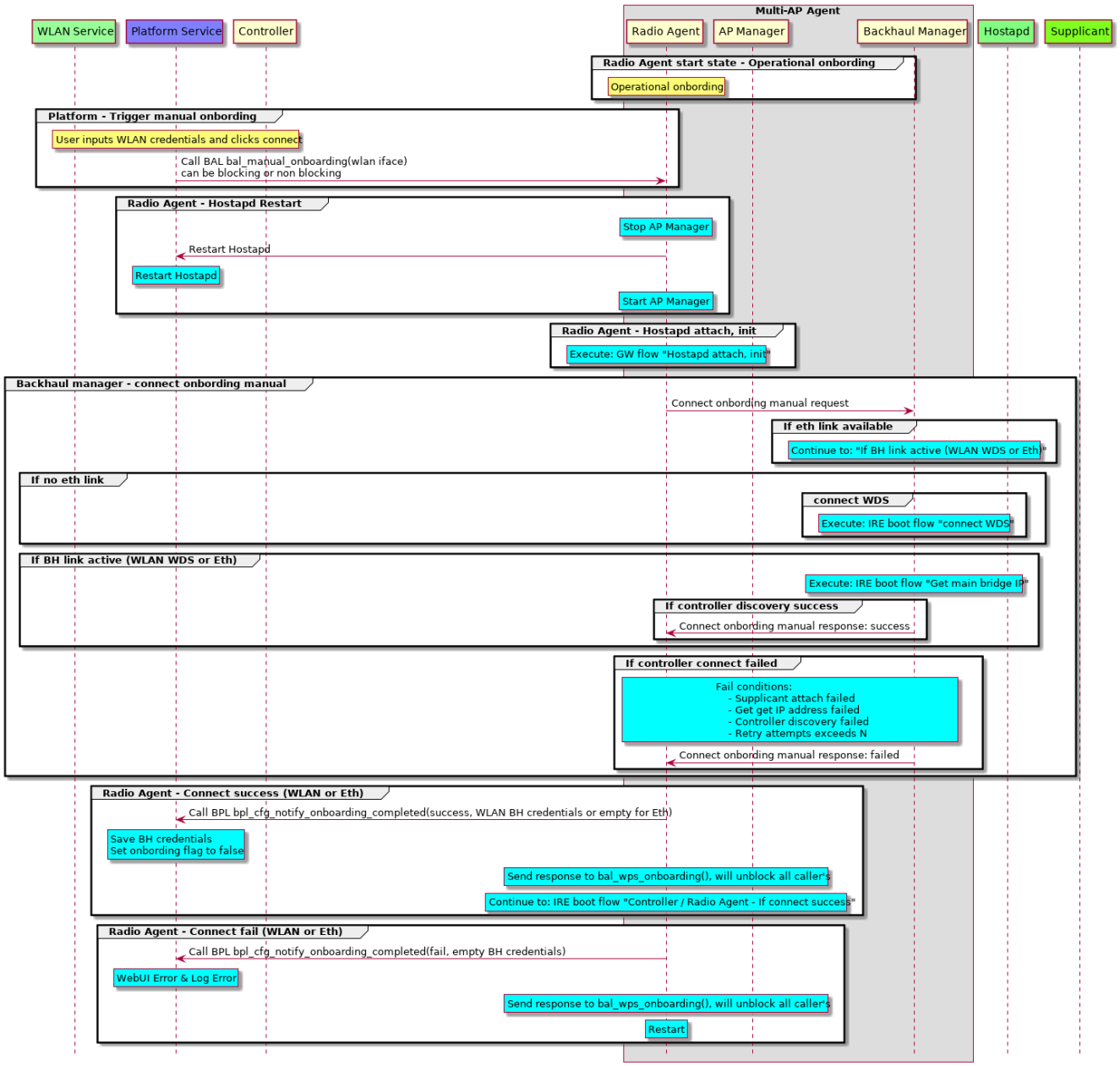

Multi-AP Agent
WLAN Service
Platform Service
Controller
Radio Agent
AP Manager
Backhaul Manager
Hostapd
Supplicant
Radio Agent - Start flow
Execute: GW flows:
- WLAN / Platform init flow
- Register to platform & backhaul manager
- Enable Hostapd
- Hostapd attach, init
IRE Backhaul manager - connect
Connect request
If eth link available
Continue to: "Get main bridge IP"
If no eth link
connect WDS
Supplicant Scan - For every registered BH radio
Start Supplicant
Start Supplicant
If supplicant is running: no action
else: start supplicant
Attach, check error
If attach timeout report error
Start Scan
Scan done
Get scan results
Select best BSSID form all bAP BSSID / SSID available from scan results (2.4G and 5G)
If SSID / bAP BSSID can't be found re-run "Supplicant Scan - For every registered BH radio"
If BH radio selected
Set Multi-AP VSIE (bSTA bit = 1)
Set 4 address mode
Add network, ssid, bssid, channel, credentials
Connect
Connected
If no BH radio selected
Stop supplicant
Stop supplicant
If BH link active (WLAN WDS or Eth)
Get main bridge IP
If DHCP client enabled
Run DHCP client on main bridge
Run DHCP client
DHCP success, bridge IP address
If DHCP client disabled
Get bridge IP address
bridge IP address
If main bridge IP valid
Execute: Multi-AP Controller Discovery
If controller discovery success
Connect response: success
If controller connect failed
Fail conditions:
- Supplicant attach failed
- Get get IP address failed
- Controller discovery failed
- Retry attempts exceeds N
Connect response: failed
If connect retry
Retry conditions:
- Scan fail
- SSID or bAP not found

## 5.4 IRE Onbording

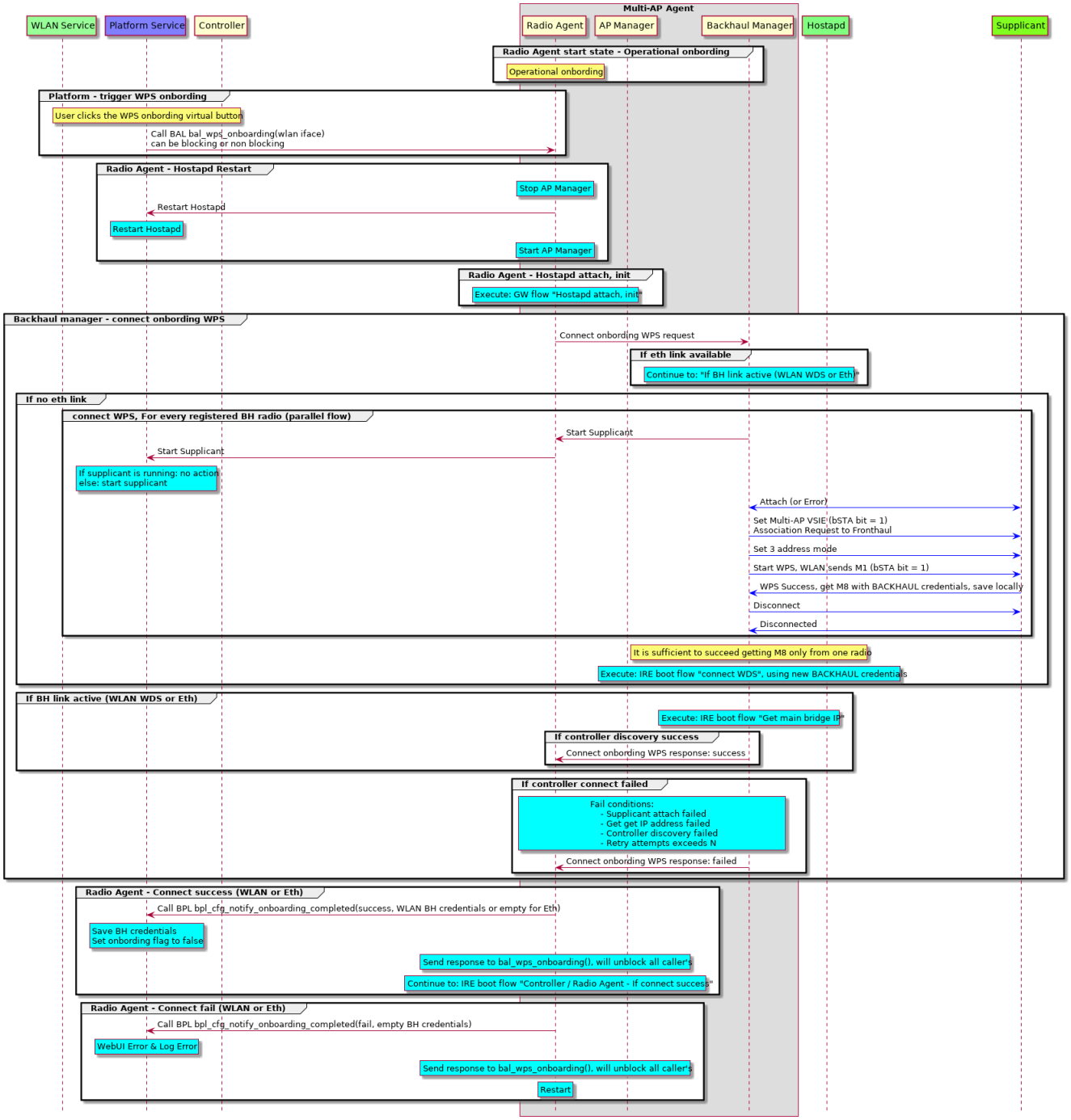
When Ethernet is configured to be used as backhaul link it is the decision of the BeeRocks backhaul manager to check the link integrity check connectivity to the controller. Ethernet backhaul check is done continuously in the background allowing the BeeRocks Agent backhaul manger switch between WLAN and Ethernet links as needed.

IRE Ethernet onboarding flow starts when platform requests agent for WPS or manual onbording. Agent automatically checks for Ethernet connection, if available Ethernet is used else agent will fall back to WLAN link.

### 5.4.1 IRE Manual Onbording



### 5.4.2 IRE WPS Onbording




## 5.5 Front / Backhaul Network Separation

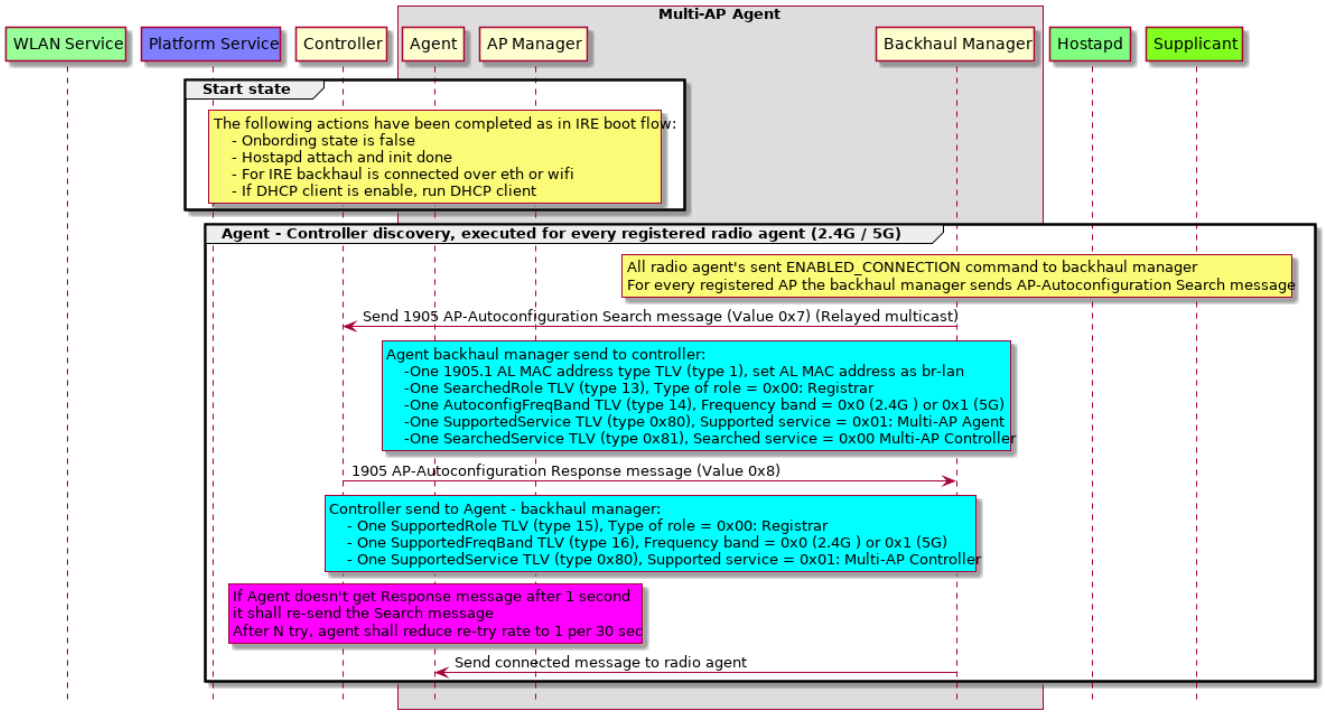
In order to resolve platform/WLAN WDS connectivity issues related to acceleration engine learning it is has been decided to use a separate network (SSID) for IRE backhaul connection.
The new connectivity schema defines a backhaul network and a front network that allows separation of 3 address station (standard clients) and WDS 4 address stations (extender) where 3 address station are only allowed to connect to the front VAP and 4 address stations are only allowed to connect to backhaul VAP. Only one extender connection is allowed per backhaul VAP i.e. BSSID. The new connection scheme moves routing logic from WLAN exclusively to platform bridge allowing platform acceleration engine to detect client port/sub-port change caused when WLAN stations steers between VAPs.

WDS Connectivity flows are defined in WDS Connectivity Application Notes.

WLAN subsystem requirements in WLAN_Requirements document, all new items are marked with V1.3.

## 5.6 Controller Discovery

Controller discovery is executed every time the agent initializes, this is relevant for GW / Repeater / Extender devices and for any agent backhaul i.e Ethernet, WiFi or bridge. As appose to other flows between agent and controller, the discovery flow is done between backhaul manager and controller. After successful completion of discovery the controller AL MAC address is known and agent can now continue to have uni-cast communication with the controller.

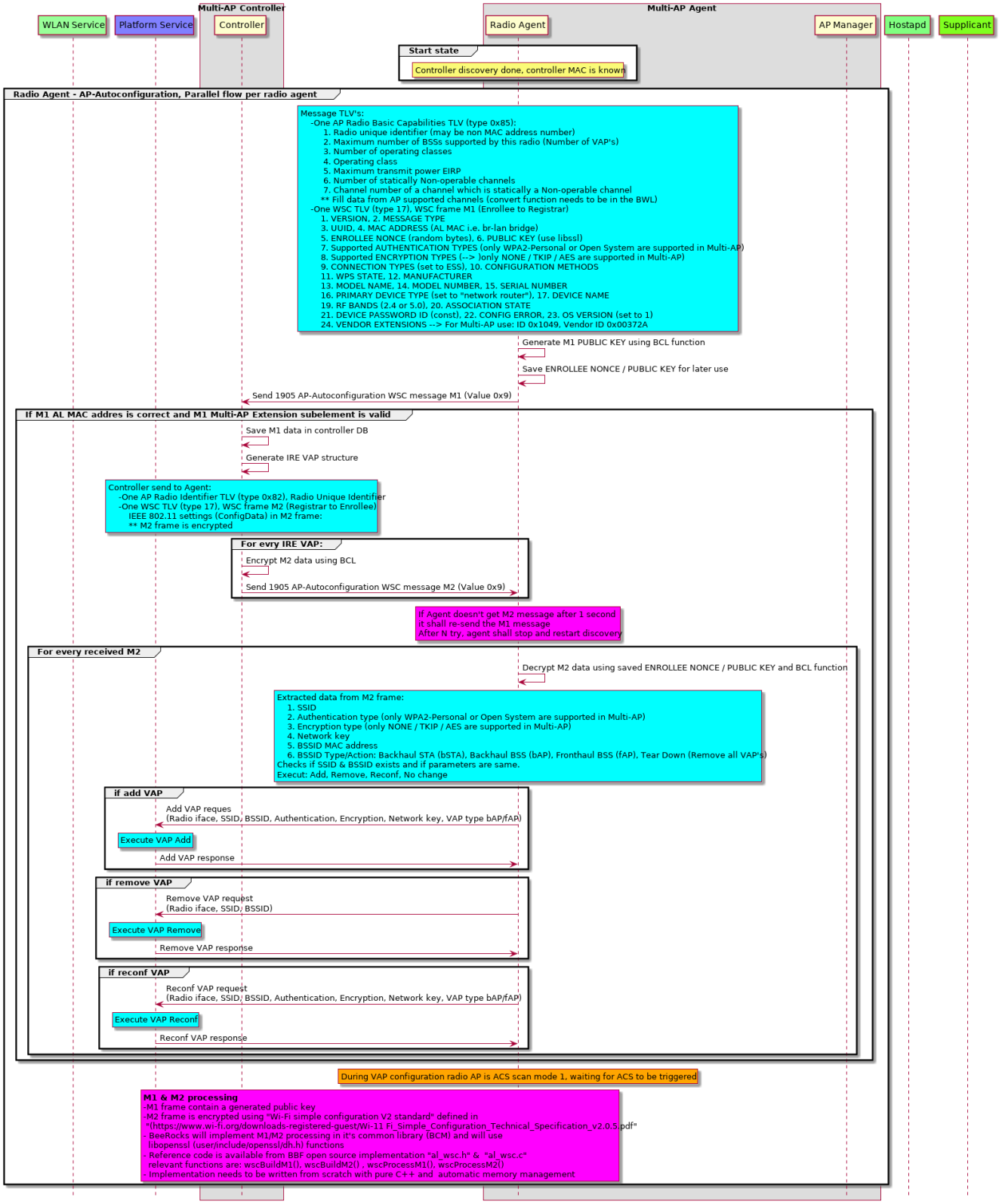



## 5.7 Multi-AP Configuration

### 5.7.1 AP Auto Configuration

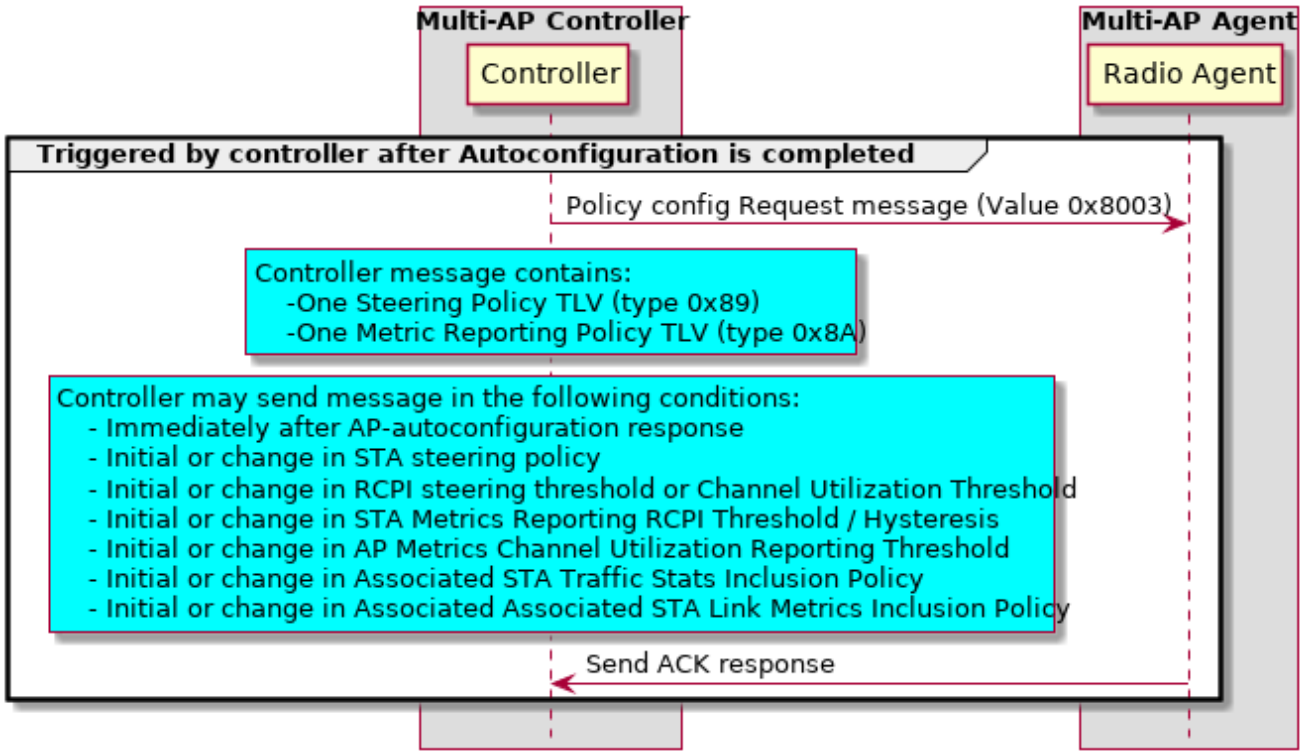
Auto configuration flow target is to configure all the SSID's and BSSID's per agent radio. Configuration include SSID name, BSSID MAC address, Authentication type, Encryption type, Network key and BSSID service type (bSTA, bBSS, fBSS). Auto configuration flow starts from radio agent after the controller discovery iscomplete. One impotent note on the flow is that the M2 message is encrypted with the public key provided in M1 message. M1 public key generation and M2 message decrepit will be done using OpenSSH

https://github.com/krzyzanowskim/OpenSSL/blob/master/include-macos/openssl/dh.h "DH" helper functions.
OpenSSH is in Intel open source white list https://whitelisttool.amr.corp.intel.com/view.php?id=4116



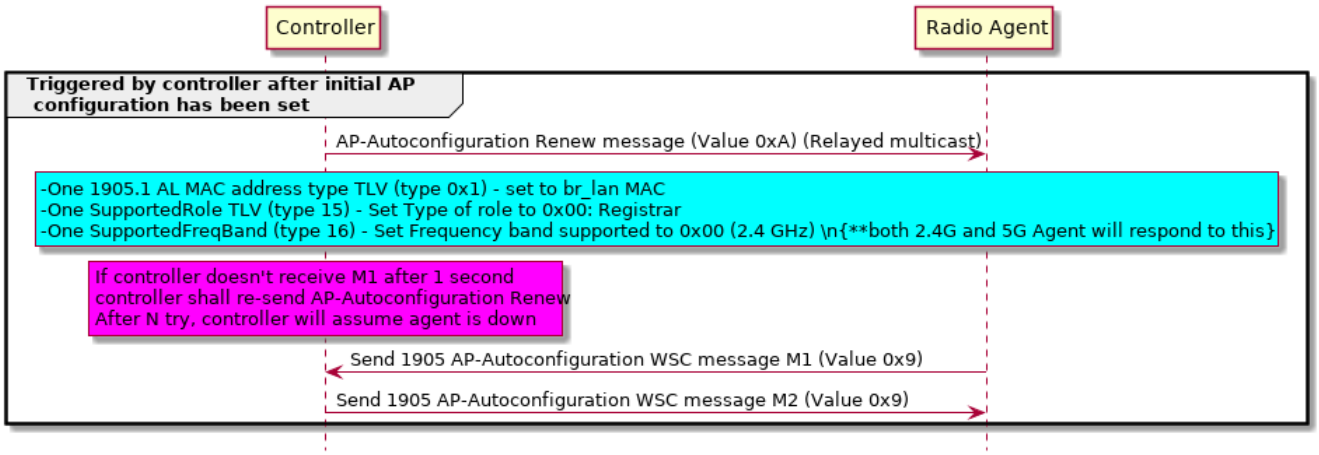
### 5.7.2 Policy Configuration

Multi-AP controller is responsible for sending policy configuration to agent at lest once after AP auto configuration response.
Controller will use read pre-defined configuration from it's configuration file and will allow run-time configuration via BML.



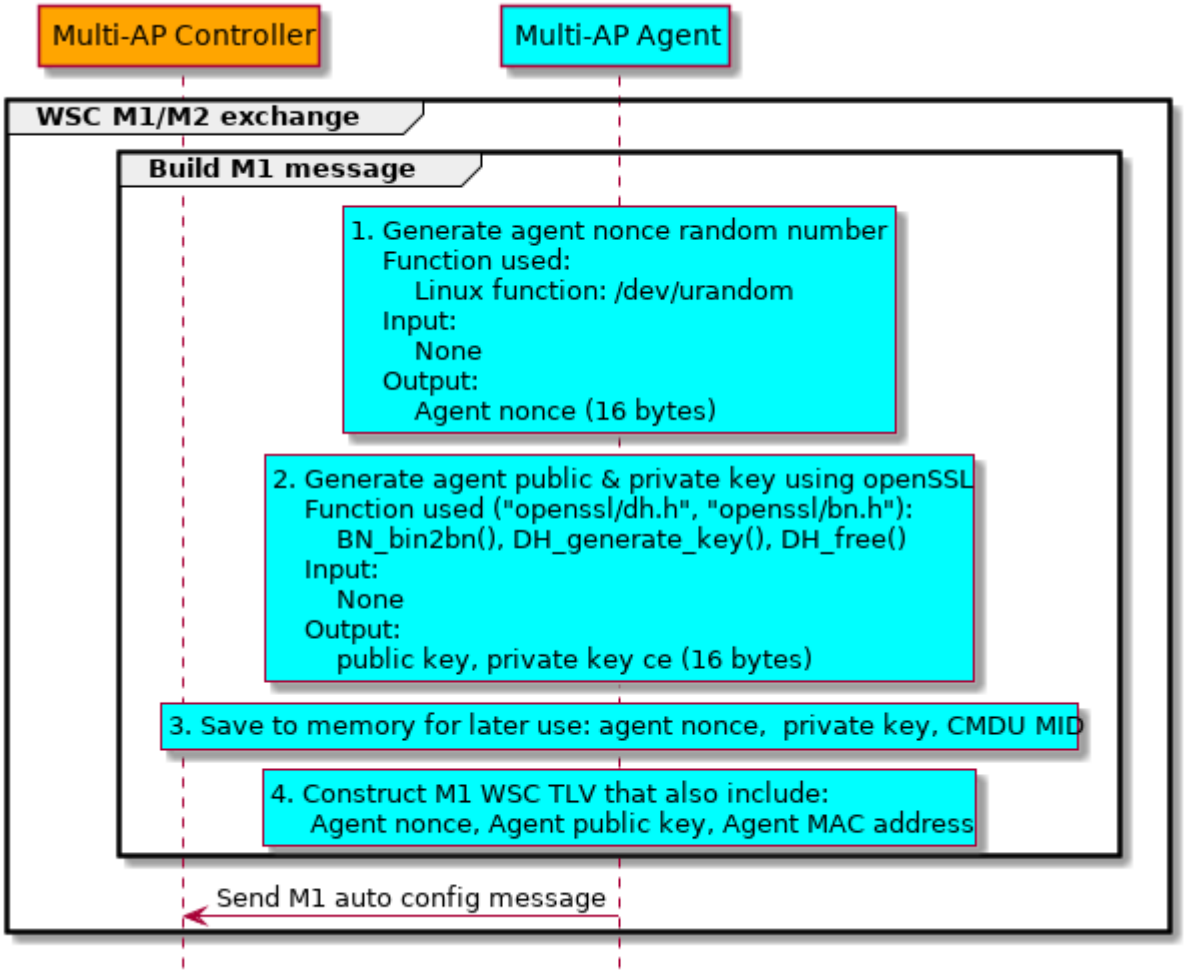

### 5.7.3 AP Auto Configuration Renew

The AP auto configuration renew flow allows the controller to request the agent to re-run the AP auto configuration flow.
This controller can send the renew message at any time during the lifetime of the agent. When received agent will start auto configuration for all agent radios i.e. 2.4G and 5G.

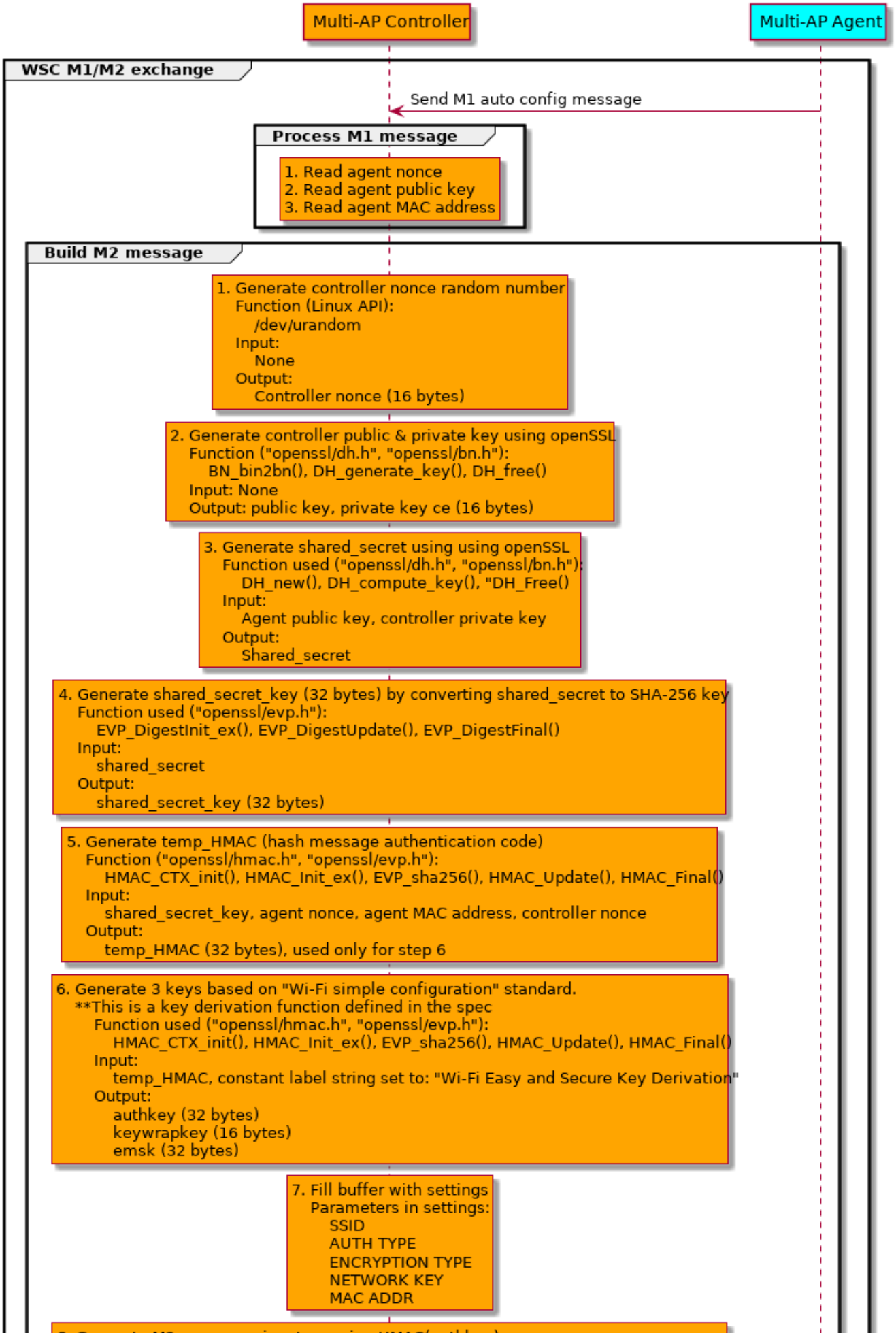


### 5.7.4 M1/2 WSC Message Encrypt/Decrypt Flows

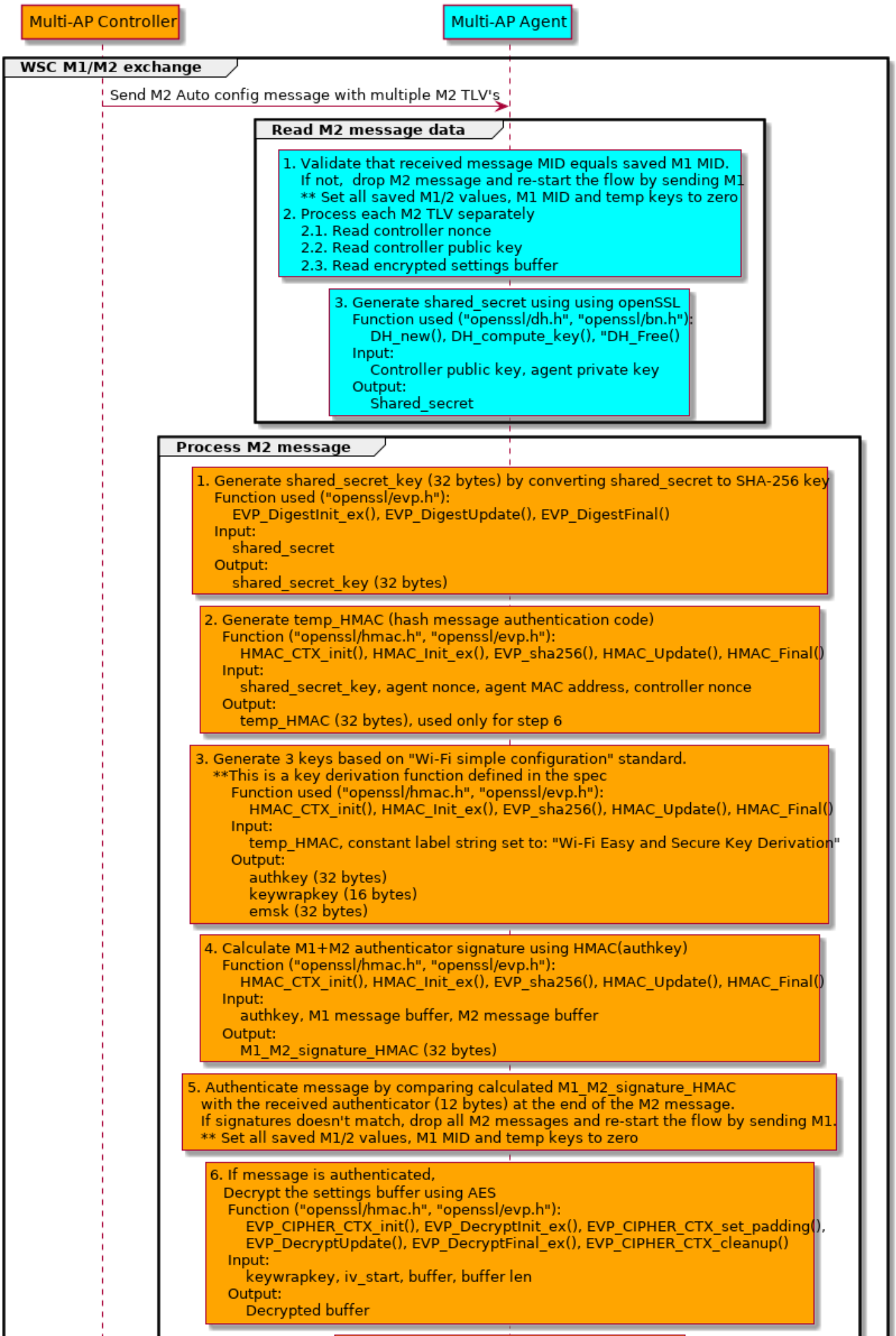
#### 5.7.4.1 Build M1 Message



#### 5.7.4.2 Process M1 Message and Build M2 Message


Multi-AP Controller
Multi-AP Agent
WSC M1/M2 exchange
Send M1 auto config message
Process M1 message
1. Read agent nonce
2. Read agent public key
3. Read agent MAC address
Build M2 message
1. Generate controller nonce random number
Function (Linux API):
/dev/urandom
Input:
None
Output:
Controller nonce (16 bytes)
2. Generate controller public & private key using openSSL
Function ("openssl/dh.h", "openssl/bn.h"):
BN_bin2bn(), DH_generate_key(), DH_free()
Input: None
Output: public key, private key ce (16 bytes)
3. Generate shared_secret using using openSSL
Function used ("openssl/dh.h", "openssl/bn.h"):
DH_new(), DH_compute_key(), "DH_Free()
Input:
Agent public key, controller private key
Output:
Shared_secret
4. Generate shared_secret_key (32 bytes) by converting shared_secret to SHA-256 key
Function used ("openssl/evp.h"):
EVP_DigestInit_ex(), EVP_DigestUpdate(), EVP_DigestFinal()
Input:
shared_secret
Output:
shared_secret_key (32 bytes)
5. Generate temp_HMAC (hash message authentication code)
Function ("openssl/hmac.h", "openssl/evp.h"):
HMAC_CTX_init(), HMAC_Init_ex(), EVP_sha256(), HMAC_Update(), HMAC_Final()
Input:
shared_secret_key, agent nonce, agent MAC address, controller nonce
Output:
temp_HMAC (32 bytes), used only for step 6
6. Generate 3 keys based on "Wi-Fi simple configuration" standard.
**This is a key derivation function defined in the spec
Function used ("openssl/hmac.h", "openssl/evp.h"):
HMAC_CTX_init(), HMAC_Init_ex(), EVP_sha256(), HMAC_Update(), HMAC_Final()
Input:
temp_HMAC, constant label string set to: "Wi-Fi Easy and Secure Key Derivation"
Output:
authkey (32 bytes)
keywrapkey (16 bytes)
emsk (32 bytes)
7. Fill buffer with settings
Parameters in settings:
SSID
AUTH TYPE
ENCRYPTION TYPE
NETWORK KEY
MAC ADDR

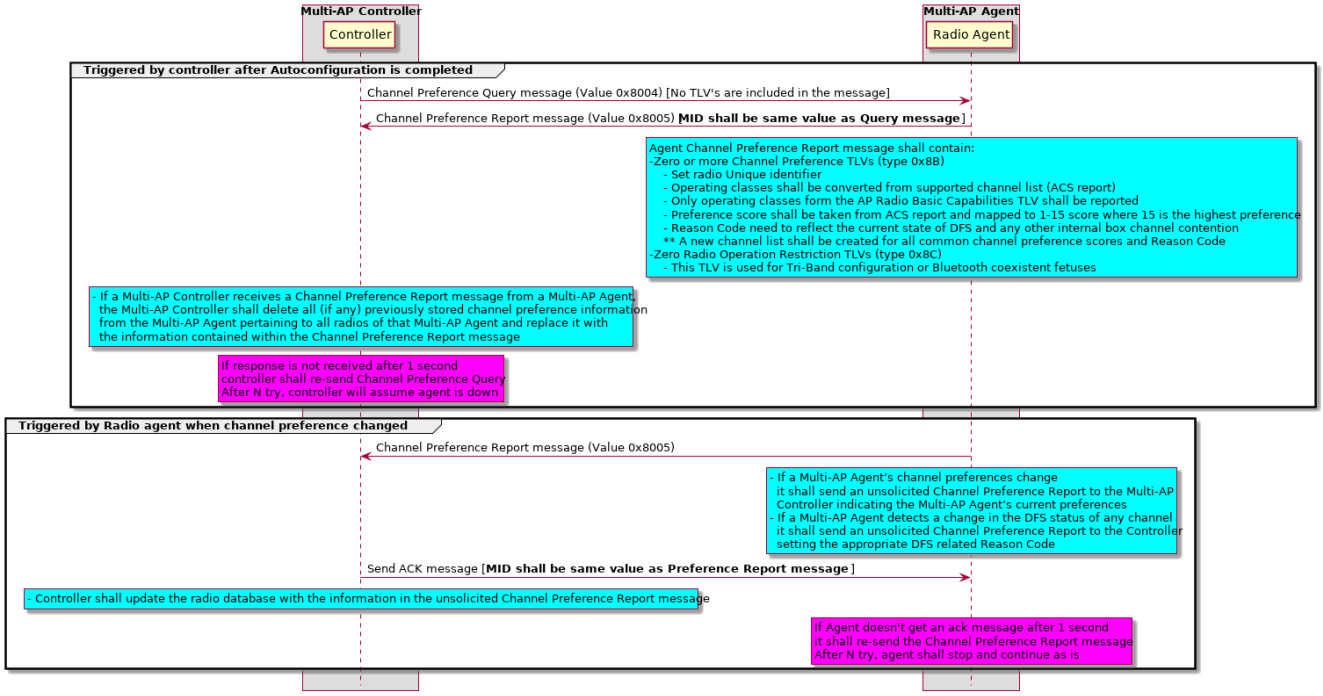
#### 5.7.4.3 Process M2 Message


Multi-AP Controller
Multi-AP Agent
WSC M1/M2 exchange
Send M2 Auto config message with multiple M2 TLV's
Read M2 message data
1. Validate that received message MID equals saved M1 MID.
If not, drop M2 message and re-start the flow by sending M1
** Set all saved M1/2 values, M1 MID and temp keys to zero
2. Process each M2 TLV separately
2.1. Read controller nonce
2.2. Read controller public key
2.3. Read encrypted settings buffer
3. Generate shared_secret using using openSSL
Function used ("openssl/dh.h", "openssl/bn.h"):
DH_new(), DH_compute_key(), "DH_Free()
Input:
Controller public key, agent private key
Output:
Shared_secret
Process M2 message
1. Generate shared_secret_key (32 bytes) by converting shared_secret to SHA-256 key
Function used ("openssl/evp.h"):
EVP_DigestInit_ex(), EVP_DigestUpdate(), EVP_DigestFinal()
Input:
shared_secret
Output:
shared_secret_key (32 bytes)
2. Generate temp_HMAC (hash message authentication code)
Function ("openssl/hmac.h", "openssl/evp.h"):
HMAC_CTX_init(), HMAC_Init_ex(), EVP_sha256(), HMAC_Update(), HMAC_Final()
Input:
shared_secret_key, agent nonce, agent MAC address, controller nonce
Output:
temp_HMAC (32 bytes), used only for step 6
3. Generate 3 keys based on "Wi-Fi simple configuration" standard.
**This is a key derivation function defined in the spec
Function used ("openssl/hmac.h", "openssl/evp.h"):
HMAC_CTX_init(), HMAC_Init_ex(), EVP_sha256(), HMAC_Update(), HMAC_Final()
Input:
temp_HMAC, constant label string set to: "Wi-Fi Easy and Secure Key Derivation"
Output:
authkey (32 bytes)
keywrapkey (16 bytes)
emsk (32 bytes)
4. Calculate M1+M2 authenticator signature using HMAC(authkey)
Function ("openssl/hmac.h", "openssl/evp.h"):
HMAC_CTX_init(), HMAC_Init_ex(), EVP_sha256(), HMAC_Update(), HMAC_Final()
Input:
authkey, M1 message buffer, M2 message buffer
Output:
M1_M2_signature_HMAC (32 bytes)
5. Authenticate message by comparing calculated M1_M2_signature_HMAC
with the received authenticator (12 bytes) at the end of the M2 message.
If signatures doesn't match, drop all M2 messages and re-start the flow by sending M1.
** Set all saved M1/2 values, M1 MID and temp keys to zero
6. If message is authenticated,
Decrypt the settings buffer using AES
Function ("openssl/hmac.h", "openssl/evp.h"):
EVP_CIPHER_CTX_init(), EVP_DecryptInit_ex(), EVP_CIPHER_CTX_set_padding(),
EVP_DecryptUpdate(), EVP_DecryptFinal_ex(), EVP_CIPHER_CTX_cleanup()
Input:
keywrapkey, iv_start, buffer, buffer len
Output:
Decrypted buffer

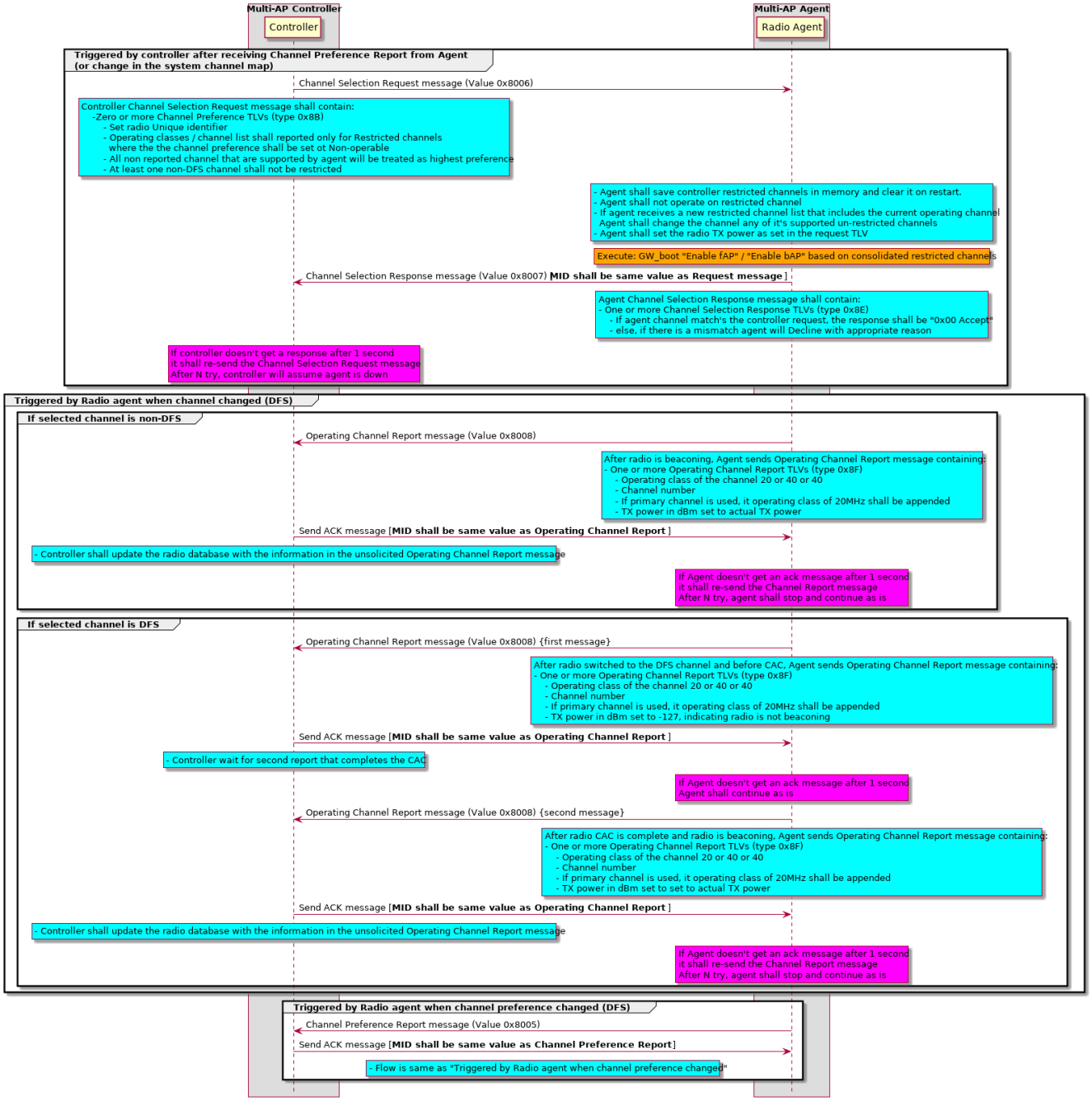
## 5.8 AP Channel Selection

Conceptually channel selection logic of Multi-AP R1.0 is similar to BeeRocks v1.2. Flows are slightly different as Multi-AP splits the flows to two parts channel preference and channel selection. Other than this difference the content of the messages is similar. Channel selection flow ends after all radios have selected a channel and are beaconing. EasyMesh uses IEEE operating classes to describe supported channels where Hostapd uses channel number / frequency and bandwidth.
Multi-AP agent is responsible for converting operating classes to operating channels and vice-versa.
Regulatory domain and operating class information is a available in the appendix section.

### 5.8.1 Channel Preference Query and Report



## 5.8.2 Channel Selection Request and Report




## 5.8.3 fAP DFS flows

When a radar is detected on an agent AP DFS channel the agent shall switch the AP to a fail-safe channel that is the non-DFS channel in the last received "channel selection request message" message (from controller). After AP switches to a non-DFS fail-safe channel, agent shall send an unsolicited operating channel report message to controller reflecting the change. Agent DFS channel re-entry may be executed in one of the following ways:

1. WAV500 or WAV600 mix band 2x2 2.4G and 2x2 5G:
   - Intel Enhancement - DFS Radar Mitigation by Client Steering (Intel controller and agent):
     Steering clients to 2.4G then clearing a new DFS channel and steering the clients back to the new 5G DFS channel.This is needed as WAV500 chip can't scan another DFS channel while keeping

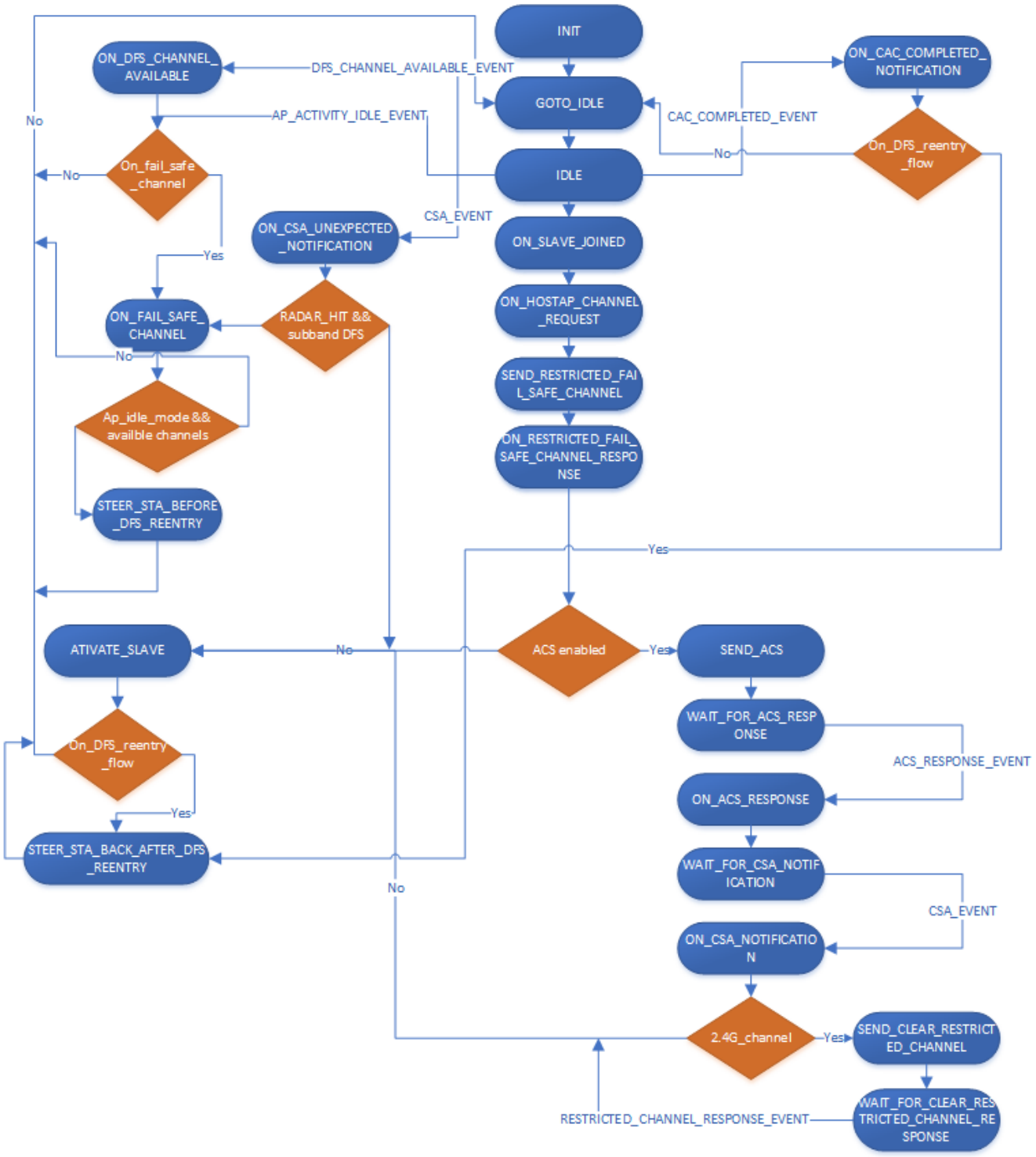
connected on the non-DFS channel. This flow is possible only with Intel enhanced flow where both controller and agent are Intel.

- Non Intel controller:
  Agent shall stay on non-DFS fail-safe channel and will not try to re-enter DFS channel.

2. WAV600 4x4:
   Agent will reduce AP number of active streams (antennas) from 4 to 3 and use the 4th free antenna and receive chain to scan a new DFS channel while keeping connected clients active with 3x3 AP (requires defecated WAV GPB).
   An unsolicited channel preference report shall be sent from agent to the controller reflecting the effected DFS channel.
   Agent will select a new DFS channel from the controller allowed list and configure the 1x1 4th antenna to radar clear the channel.Once the new channel is cleared the agent will switch the AP to 4x4 operation on the new DFS channel. An unsolicited channel preference report shall be sent to the controller reflecting the new operating channel.
   If during the CAC period (DFS channel radar scan) a new "Channel Selection Request" is received from controller that conflicts with the current selected DFS channel, the agent shall stop it's operation and re-select a channel based on controller preference.

#### 5.8.3.1 Intel Enhancement - DFS Radar Mitigation by Client Steering

The following flow chart describes the Intel DFS radar mitigation by steering the connected 5G clients to 2.4G band while a new DFS channel is cleared then steering the clients back.




Notes:

1. Relevant when operational on a 5GHz radio with a DFS channel.
2. A radar is detected on the operating channel.
3. The WLAN driver changes the radio's channel to a fail-safe channel.
4. A CSA event notification is received, with the newly selected fail-safe channel.
5. If AP activity mode is IDLE and 80MHz DFS channels are available, DFS reentry will take place.
6. If the conditions from step 5 above are not met, the radio will remain in the fail-safe channel until the conditions are met.
7. Reentry flow :

1. All stations connected to the 5GHz Radio are steered to the 2.4GHz radio.
2. ACS is triggered.
3. ACS/CSA is received.
4. If a DFS channel is selected waiting for CAC complete.
5. Slave is activated.
6. All steered stations client from step i) are steered back to 5G radio.

### 5.8.4 Repeater / Extender bSAT Channel Switch

1. bSAT (i.e. station on repeater / extender side) shall support 11h channel switch, such that when AP switches a channel with 11h the station shall switch channel together with the AP without disconnecting.
2. If bSTA is sharing the radio with a fAP/bAP and the channel switch will interfere with AP operation (mainly 5G radio), the agent shall disconnect the bSAT and re-connect using the other band (2.4G).

### 5.8.5 Intel Sub-band Radar-hit DFS Flow

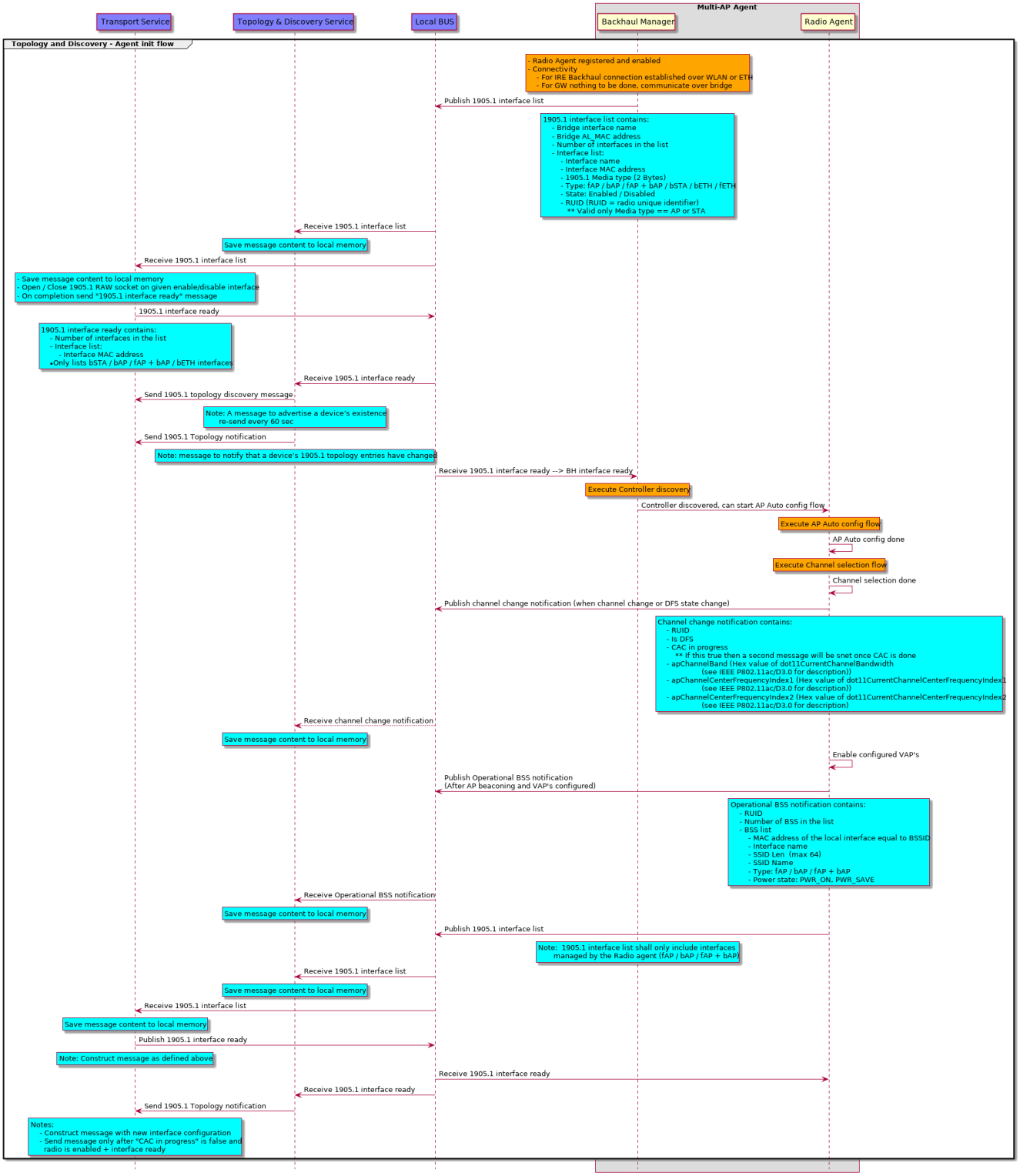
1. Sub-band DFS allows the AP to change bandwidth and primary channel while keeping the PHY open on a the same 80MHz channel.
2. For every sub-band DFS event Hostapd will send CSA with the new channel / bandwidth to AP manager that propagate to the agent. As a response the agent will generate an unsolicited "Operating Channel Report" message reflecting the defected channels (20MHz slices).
3. According to configuration the controller may or may not allow the agent to remain on the new reduced channel / bandwidth.
4. For platforms running WAV600 4x4, agent may be configured to execute DFS re-entry when the bandwidth drops below configured threshold. DFS re-entry is done using the 4th antenna in DFS mode.

## 5.9 1905.1 Topology and Discovery - Agent flows

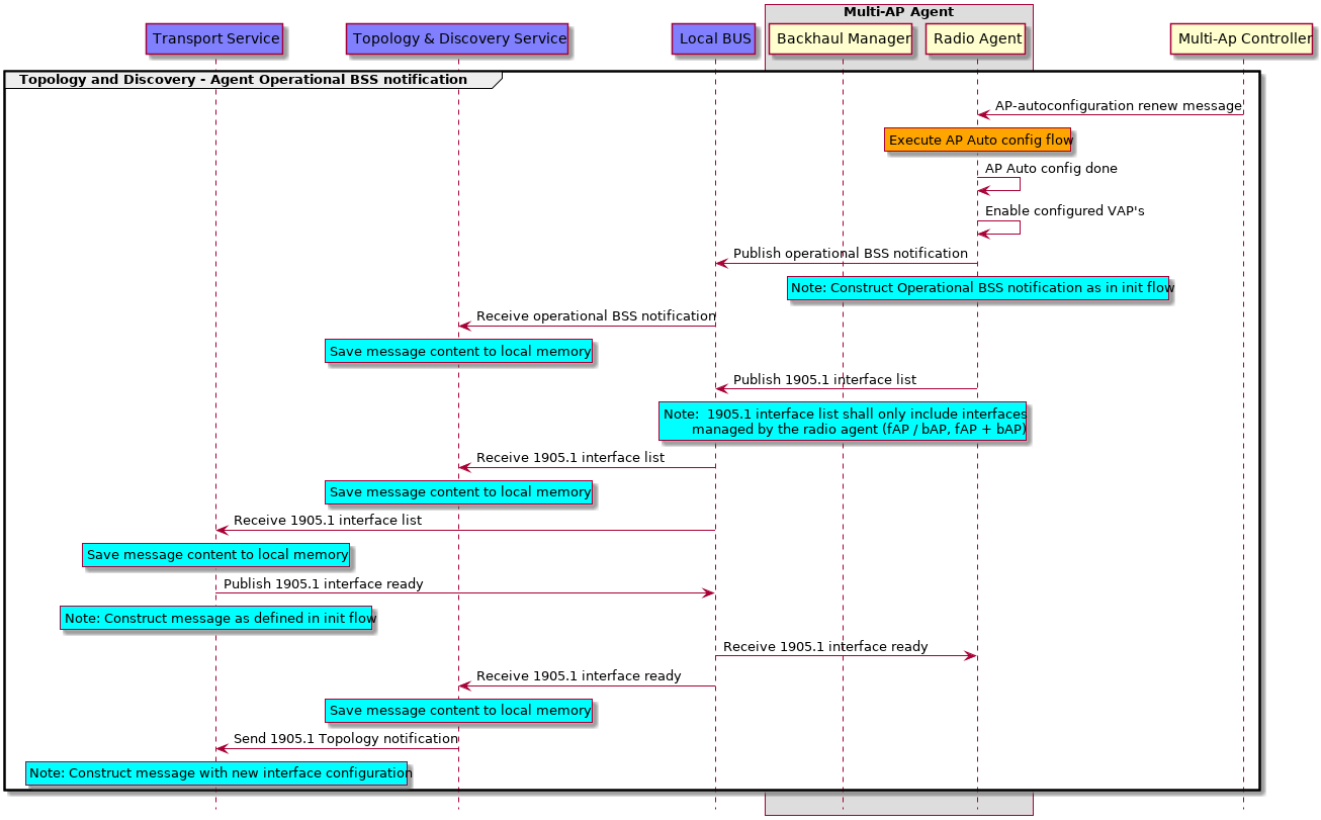
In the Multi-AP framework architecture it is the role of the Topology and discovery Service is responsible for sending
1905.1 Discovery notification, 1905.1 Topology notification and 1905.1 Topology response messages.
It is the job of the Multi-AP agent to notify the topology and discovery service with the relevant information on the active BSS list, operating channel, clients connected and backhaul interface.

The following sections provide high level flows between the entities.

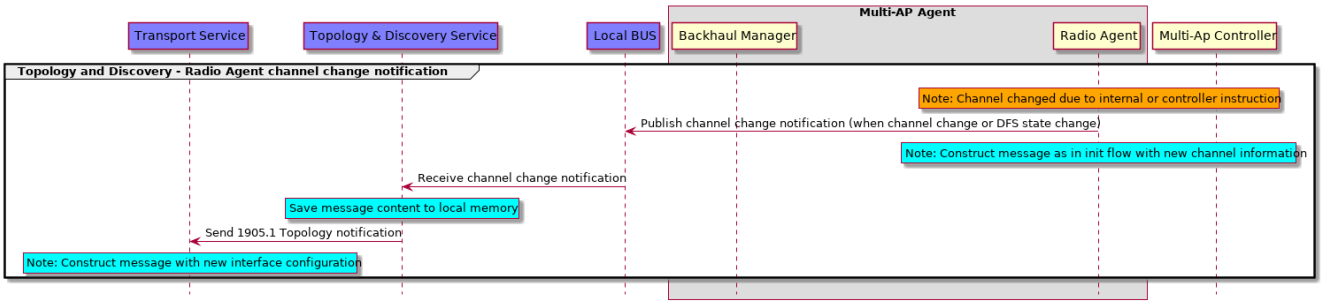
### 5.9.1 Topology and Discovery Init Flow



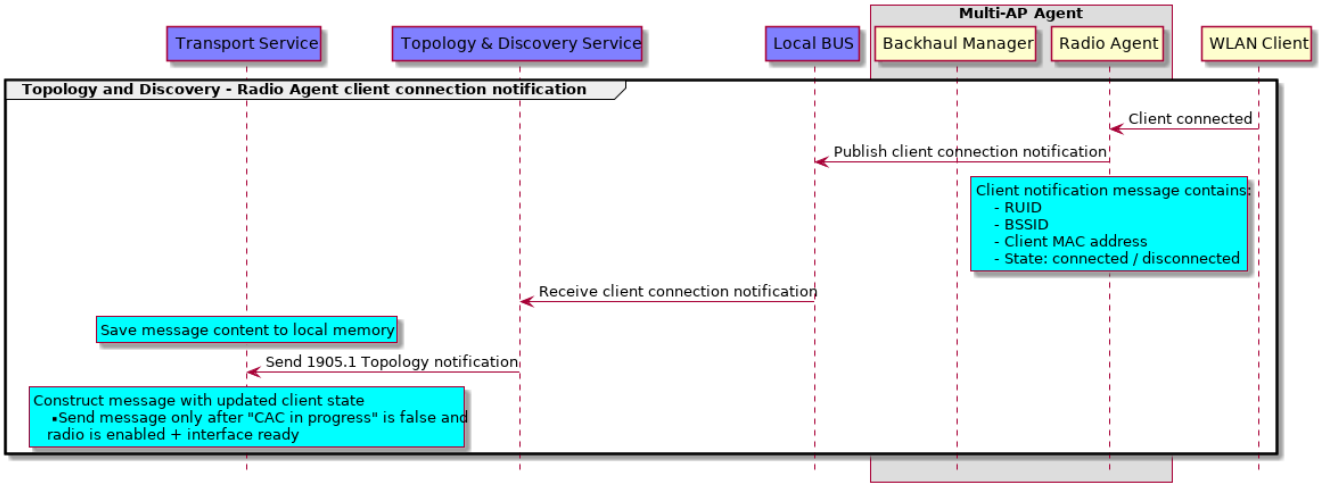

### 5.9.2 Topology and Discovery Operational BSS Notification



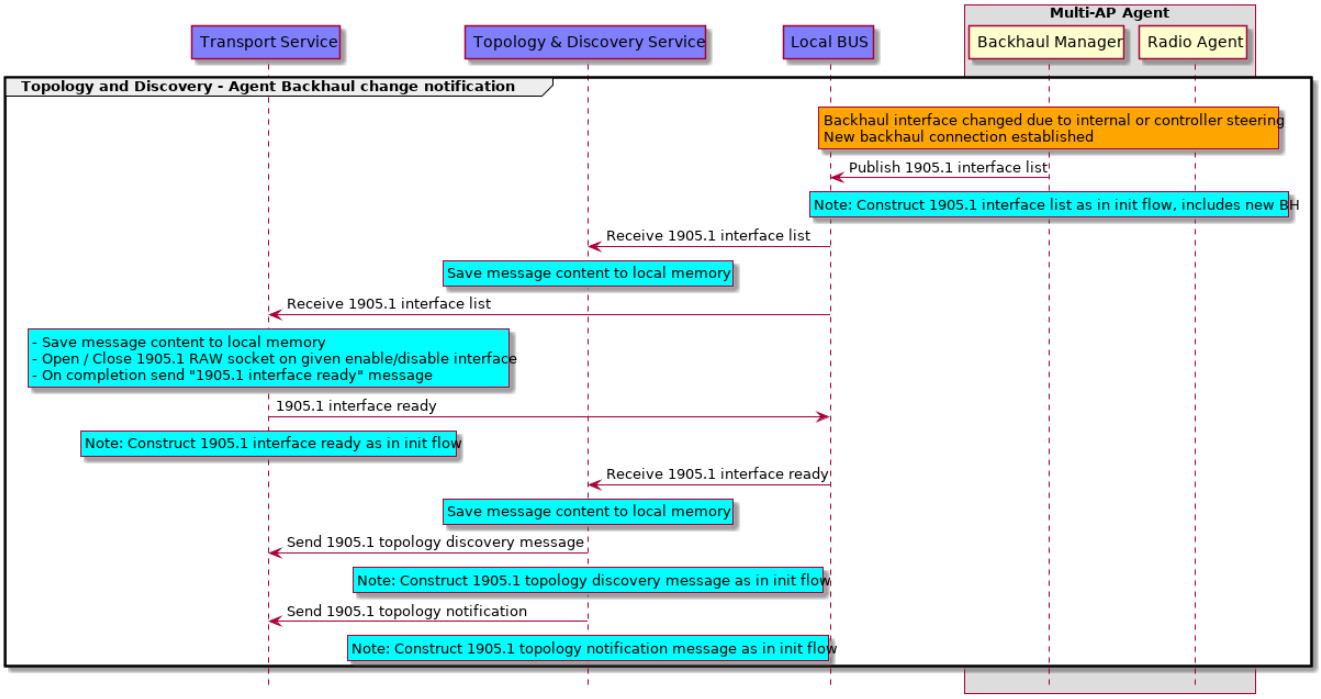

## 5.9.3 Topology and Discovery Channel Change Notification



### 5.9.4 Topology and Discovery Client Connection Notification




### 5.9.5 Topology and Discovery Backhaul Change Notification




## 5.10 Capability Information Reporting

### 5.10.1 AP capability

### 5.10.2 Client capability

## 5.11 Link Metric Collection

### 5.11.1 Backhaul Link Metrics

### 5.11.2 Per-AP Link Metrics

### 5.11.3 Link Metric Measurements from the AP

### 5.11.4 Per-STA measurements

#### 5.11.4.1 Associated STA Link Measurements From the AP

#### 5.11.4.2 Unassociated STA RCPI Measurements From the AP

#### 5.11.4.3 11K Beacon Measurements From the Client

#### 5.11.4.4 Combined Infrastructure Metrics

## 5.12 Client steering

### 5.12.1 Multi-AP Controller initiated steering mandate

### 5.12.2 Multi-AP Controller initiated steering opportunity

### 5.12.3 Multi-AP Agent initiated RCPI-based steering

### 5.12.4 Legacy Client steering

## 5.13 Backhaul Optimization (Backhaul SON)

## 5.14 Higher layer data payload over 1905

# 6 Appendix

## 6.1 Regulatory Domain

| Geographic area | Approval standards | Documents | Approval authority |
|---|---|---|---|
| Canada | Minister of Industry | RSS-210 — Licence-exempt Radio Apparatus (All Frequency Bands): Category I Equipment | Industry Canada |
| China | Ministry of Industry and Information Technology (MIIT) | Xin Bu Wu [2002] #353, Xin Bu Wu [2002] #277, Gong Xin Bu Wu Han [2012] #620 | MIIT |
| Europe | European Conference of Postal and Telecommunications (CEPT) -Administrations and its Electronic Communications Committee (ECC). Also, European Radiocommunications Office, European Telecommunications - Standards Institute (ETSI) | ECC DEC (04) 08, ETSI EN 300 328 [B13], ETSI EN 301 893, ETSI ES 202 663 [B15], ETSI EN 302 571 [B14], Clause 5 | CEPT |
| Japan | Ministry of Internal Affairs and Communications (MIC) | MIC Equipment - Ordinance (EO) for Regulating Radio Equipment Articles 7, 49.20, 49.21 | MIC |
| United States | Federal Communications - Commission (FCC) | 47 CFR [B9], Part 15, Sections 15.205, 15.209, and15.247 and 15.255; and Subpart E, Sections 15.401–15.407, and Subpart H, Sections 15.701–15.716, Section 90.210, Sections 90.371–383, Sections 90.1201–90.1217, Sections 90.1301–90.1337, Section 95.639, Sections 95.1501–1511 | FCC |

## 6.2 Operating Class

The operating class is an index into a set of values for radio operation in a regulatory domain.

The channel starting frequency variable is a frequency, used together with an operating class number and a channel number, to calculate a channel center frequency.For china the channel starting frequency is not defined by the operating class and is derived from regulation. Channel spacing is the frequency difference between non overlapping adjacent channel center frequencies when using the maximum bandwidth of one frequency segment allowed for this operating class.

The channel set is the list of integer channel numbers that are legal for a regulatory domain and class. A “—” in the Channel set column of the operating classes tables indicates either that the values in the channel center frequency index field apply for calculating channel center frequencies of this operating class, or where both the channel set field and the channel center frequency index field are “—” indicates that the channel set is not defined by the operating class and is derived from regulation.

The channel center frequency index is the set of integer channel numbers that correspond to frequency segments and that are allowed for the operating class. A“—”in the Channel center frequency index column of the operating classes tables indicates either that the values in the channel set field apply for calculating channel center frequencies of this operating class or, when both the Channel set and the Channel center frequency index column of the operating classes tables are “—”,indicates that the channel center frequency index is not defined by the operating class and is derived from regulation.

**Global operating classes**

| Operating class | Nonglobal operating class(es) | Channel starting frequency (GHz) | Channel spacing (MHz) | Channel set | Channel center frequency index | Behavior limits set |
|---|---|---|---|---|---|---|
| 1–80 | — | Reserved | Reserved | Reserved | — | Reserved |
| 81 | E-1-12, E-2-4, E-3-30, E-5-7 | 2.407 | 25 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | — | — |
| 82 | E-3-31 | 2.414 | 25 | 14 | — | — |
| 83 | E-1-32, E-2-11, E-3-56, E-5-8 | 2.407 | 40 | 1, 2, 3, 4, 5, 6, 7, 8, 9 | — | PrimaryChannelLowerBehavior |
| 84 | E-1-33, E-2-12, E-3-57, E-5-9 | 2.407 | 40 | 5, 6, 7, 8, 9, 10, 11, 12, 13 | — | PrimaryChannelUpperBehavior |
| 85 | — | — | 6, 7, 8 | — | — | GeoDB |
| 86 | — | — | 12, 14, 16 | — | — | GeoDB |

| | | | | | | |
|---|---|---|---|---|---|---|
| 87 | — | — | 24, 28, 32 | — | — | GeoDB |
| 88-93 | — | Reserved | Reserved | Reserved | Reserved | Reserved |
| 94 | E-1-13 | 3 | 20 | 133, 137 | — | CCA-EDBehavior |
| 95 | E-1-14 | 3 | 10 | 132, 134, 136, 138 | — | CCA-EDBehavior |
| 96 | E-1-15 | 3.0025 | 5 | 131, 132, 133, 134, 135, 136, 137, 138 | — | CCA-EDBehavior |
| 97–100 | — | Reserved | Reserved | Reserved | Reserved | Reserved |
| 101 | E-1-10,11 | 4.85 | 20 | 21, 25 | — | — |
| 102 | E-1-8,9 | 4.89 | 10 | 11, 13, 15, 17, 19 | — | — |
| 103 | E-1-6,7 | 4.9375 | 5 | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 | — | — |
| 104 | E-3-46,47,48,49,50 | 4 | 40 | 184, 192 | — | PrimaryChannelLowerBehavior |
| 105 | E-3-51,52,53,54,55 | 4 | 40 | 188, 196 | — | PrimaryChannelUpperBehavior |
| 106 | — | 4 | 20 | 191, 195 | — | — |
| 107 | — | 4 | 10 | 189, 191, 193, 195, 197 | — | — |
| 108 | — | 4.0025 | 5 | 188, 189, 190, 191, 192, 193, 194, 195, 196, 197 | — | — |
| 109 | E-3-7,8,9,10,11 | 4 | 20 | 184, 188, 192, 196 | — | — |
| 110 | E-3-16,17,18,19,20 | 4 | 10 | 183, 184, 185, 186, 187, 188, 189 | — | — |

| | | | | | | |
|---|---|---|---|---|---|---|
| 111 | E-3-25,26,27,28,29 | 4.0025 | 5 | 182, 183, 184, 185, 186, 187, 188, 189 | — | — |
| 112 | E-3-2,3,4,5,6 | 5 | 20 | 8, 12, 16 | — | — |
| 113 | E-3-12,13,14,15 | 5 | 10 | 7, 8, 9, 10, 11 | — | — |
| 114 | E-3-21,22,23,24 | 5.0025 | 5 | 6, 7, 8, 9, 10, 11 | — | — |
| 115 | E-1-1, E-2-1, E-3-1, E-5-1 | 5 | 20 | 36, 40, 44, 48 | — | UseEirpForVHTTxPowEnv |
| 116 | E-1-22, E-2-5, E-3-36, E-5-4 | 5 | 40 | 36, 44 | — | PrimaryChannelLowerBehavior, UseEirpForVHTTxPowEnv |
| 117 | E-1-27, E-2-8, E-3-41 | 5 | 40 | 40, 48 | — | PrimaryChannelUpperBehavior, UseEirpForVHTTxPowEnv |
| 118 | E-1-2, E-2-2, E-3-32,33, E-5-2 | 5 | 20 | 52, 56, 60, 64 | — | DFS_50_100_Behavior , UseEirpForVHTTxPowEnv |
| 119 | E-1-23, E-2-6, E-3-37,38, E-5-5 | 5 | 40 | 52, 60 | — | PrimaryChannelLowerBehavior, DFS_50_100_Behavior, UseEirpForVHTTxPowEnv |
| 120 | E-1-28, E-2-9, E-3-42,43 | 5 | 40 | 56, 64 | — | PrimaryChannelUpperBehavior, DFS_50_100_Behavior, UseEirpForVHTTxPowEnv |
| 121 | E-1-4, E-2-3, E-3-34,35,58 | 5 | 20 | 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140 | — | DFS_50_100_Behavior, UseEirpForVHTTxPowEnv |
| 122 | E-1-24, E-2-7, E-3-39,40 | 5 | 40 | 100, 108, 116, 124, 132 | — | PrimaryChannelLowerBehavior, DFS_50_100_Behavior, UseEirpForVHTTxPowEnv |

| | | | | | | |
|---|---|---|---|---|---|---|
| 123 | E-1-29,<br>E-2-10,<br>E-3-44,45 | 5 | 40 | 104, 112, 120, 128, 136 | — | PrimaryChannelUpperBehavior, DFS_50_100_Behavior, UseEirpForVHTTxPowEnv |
| 124 | E-1-3 | 5 | 20 | 149, 153, 157, 161 | — | NomadicBehavior, UseEirpForVHTTxPowEnv |
| 125 | E-1-5,<br>E-2-17,<br>E-5-3 | 5 | 20 | 149, 153, 157, 161, 165, 169 | — | LicenseExemptBehavior, UseEirpForVHTTxPowEnv |
| 126 | E-1-25,26,<br>E-5-6 | 5 | 40 | 149, 157 | — | PrimaryChannelLowerBehavior, UseEirpForVHTTxPowEnv |
| 127 | E-1-30,31 | 5 | 40 | 153, 161 | — | PrimaryChannelUpperBehavior, UseEirpForVHTTxPowEnv |
| 128 | E-1-128,<br>E-2-128,<br>E-3-128<br>E-5-128 | 5 | 80 | — | 42, 58, 106, 122, 138, 155 | UseEirpForVHTTxPowEnv |
| 129 | E-1-129,<br>E-2-129,<br>E-3-129<br>E-5-129 | 5 | 160 | — | 50, 114 | UseEirpForVHTTxPowEnv |
| 130 | E-1-130,<br>E-2-130,<br>E-3-130<br>E-5-130 | 5 | 80 | — | 42, 58, 106, 122, 138, 155 | 80+, UseEirpForVHTTxPowEnv |
| 131–179 | — | Reserved | Reserved | Reserved | Reserved | Reserved |
| 180 | E-1-34,<br>E-2-18,<br>E-3-59 | 56.16 | 2160 | 1, 2, 3, 4, 5, 6 | — | — |
| 181–191 | — | Reserved | Reserved | Reserved | — | Reserved |
| 192–254 | — | Vendor specific | Vendor specific | Vendor specific | — | Vendor specific |
| 255 | — | Reserved | Reserved | Reserved | — | Reserved |

NOTE 1—The channel spacing for operating classes 116, 117, 119, 120, 122, 123, 126, and 127 specifies the maximum radio bandwidth of one frequency segment In these operating classes, the AP operates in a 20/40 MHz BSS, and the operating channel width for a non-AP STA is either 20 MHz or 40 MHz.
NOTE 2—The channel spacing for operating classes 128, 129, and 130 specifies the maximum radio bandwidth of one frequency segment.

**Table D-2 – Behavior limits set**

| Encoding | Behavior limits set | Description |
|---|---|---|
| 19 | 80+ | In an channel bandwidth that contains two or more frequency segments, the frequency segment that does not contain the primary 80 MHz channel (see NOTE 2) |
| 20 | UseEirpForVhtTxPowEnv | A STA that sends one or more a VHT Transmit Power Envelope elements shall indicate EIRP in the Local Maximum Transmit Power Units Interpretation subfield in one of the VHT Transmit Power Envelope elements |
| 1921-255 | Reserved | Reserved |

NOTE1—The fields that specify the 40 MHz channels are described in 20.3.15.4.
NOTE 2—For an example using an operating class with an 80+ Behavior limit, see 8.4.2.10.

# 7 Abbreviations, Terminology and Terminology

| Acronym | Term |
|---|---|
| **AP** | Access Point |
| **API** | Application Programming Interface |
| **FW** | Firmware |
| **DB** | Database |
| **GW** | Gateway |
| **IRE** | Intelligent Range Extender |
| **DFS** | Dynamic Frequency Selection |
| **QoS** | Quality Of Service |
| **RRM** | Radio Resource Manager |
| **SON** | Self-Optimizing Network / Self-Organizing Network |
| **ARP** | Address Resolution Protocol |
| **DITIM** | Delivery Traffic Indication Message |
| **RTS** | Reedy To Send |
| **CTS** | Clear To Send |
| **VAP** | Virtual Access Point |
| **IP** | Internet Protocol |
| **IPC** | Inter Process Communication |
| **QoS** | Quality Of Service |
| **QoE** | Quality Of Experience |
| **RADIUS** | Remote Authentication Dial In User Service |
| **SDK** | Software Development Kit |
| **TCP** | Transmission Control Protocol |
| **UE** | User Equipment |
| **UI** | User Interface |
| **UX** | User Experience |
| **LvP** | Links View Park |
| **WAN** | Wide Area Network |
| **WWAN** | Wireless Wide Area Network |
| **WLAN** | Wireless Local Area Network |
| **WPA** | WiFi Protected Access |
| **WFA** | WiFi Alliance |

| | |
|---|---|
| **VAP** | Virtual Access Point |
| **sVAP** | station VAP |
| **bVAP** | backhaul VAP |
| **fVAP** | front VAP |