AN14996

Integrating Wi-Fi EasyMesh in a Yocto Environment

Rev. 1.0 — 28 April 2026 Application note

Document information

Information	Content
Keywords	AN14996, Wi-Fi, EasyMesh, FRDM-IMX93, Yocto, meta-matter, Controller, Agent, Multi-AP, Internet routing
Abstract	This application note provides guidance on integrating the NXP Wi-Fi EasyMesh software into a Yocto build environment containing the meta-matter BSP. It details the generalized procedure to configure, bring up, and test a general-purpose multi-AP network on FRDM-IMX93 hardware with onboard IW612 Wi-Fi 6 + Bluetooth 5.4 connectivity chip, ensuring fronthaul end devices have full Internet access.

1 Introduction

Wi-Fi EasyMesh is a Wi-Fi Alliance standard that extends the flexibility and interoperability of a Wi-Fi network. Instead of a traditional architecture where an access point (AP) manages a network of multiple stations (STA), EasyMesh uses a multi-AP controller to manage a network of multiple APs called agents. As a result, different multi-APs can work together and expand the coverage and efficiency.

1.1 Supported products

The following products support the EasyMesh feature:

- 88Q9098 (ref.[1])
- 88W9098 (ref.[2])
- AW611 (ref.[3])
- IW611 (ref.[4])
- IW612 (ref.[5])
- IW610 (ref.[6])

2 EasyMesh network topology

A typical Wi-Fi network comprises an access point (AP) along with one or more STAs. The limited coverage of an individual AP can be expanded through Multi-AP networks. EasyMesh is one of the standardized ways to implement a multi-AP network. To construct the network, EasyMesh uses one multi-AP controller and some multi-AP agents. The network can be expanded in various configurations, depending on the user's deployment topology.

An EasyMesh network supports up to ten agents, featuring automatic loop detection, prevention, and self-healing capabilities. Onboarding new agents requires a simple button push or QR code.

2.1 Multi-AP controller

The Controller is used to:

• Control the fronthaul and backhaul links of the network.

• Receive and monitor the measurements and capability data.

• Onboard the agents.

When the Controller is down, the EasyMesh management plane becomes unavailable. While agents can maintain backhaul connectivity, clients cannot access the Internet and network orchestration is suspended.

Note: *Multi-AP controller is later referred to as Controller.*

2.2 Multi-AP agents (later referred to as Agent)

One or more agents:

- Execute the commands received from the controller.
- Report the measurements and capabilities data to the controller or to another agent.

An agent consists of:

- Fronthaul AP (fAP or fBSS): Used for client STAs to associate with the AP for Wi-Fi connectivity:
  - It operates as an AP to apply configuration changes and execute AP control functions.

  - It allows client STAs to associate with Wi-Fi connectivity.
- Backhaul AP (bAP or bBSS): Being part of Wi-Fi backhaul connectivity enables the backhaul STAs to associate directly with the backhaul AP:
  - It enables reliable communication between mesh nodes.
  - It does not serve client devices directly; instead maintains the mesh topology and forwards data between agents.
- A combo BSS is when one agent runs a single BSS instead of separate fBSS and bBSS. In the EasyMesh Yocto recipe and EasyMesh in Matter BSP, the controller is configured for Combo BSS mode by default, that is, the same uap0 interface acts as both fronthaul and backhaul BSS. To use a separate BSS configuration, refer to Section 6.4 "Advanced configuration: Separate fronthaul and backhaul BSS".
- Backhaul STA (bSTA): As part of Wi-Fi backhaul connectivity, the STA associates with parent agent backhaul AP:
  - It operates as a STA to get measurements and capabilities data associated with backhaul AP, for backhaul connectivity.

A bSTA and bBSS always communicate in a 4-address mode.

2.3 EasyMesh network topology formations

This section includes different EasyMesh network setups that illustrate how the controller/agents connect in various topologies. The visuals highlight the flexibility of EasyMesh in extending and optimizing Wi-Fi coverage.

Figure 1 shows an Agent connected to the Controller via Ethernet.

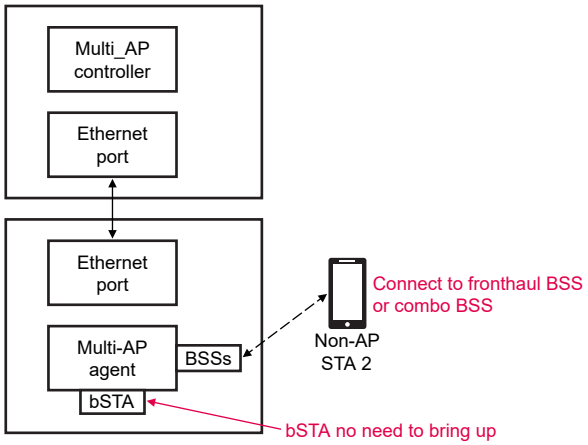

Figure 1. An Agent connected to the Controller via Ethernet

Figure 2 shows an agent hosted on the local device as the controller.

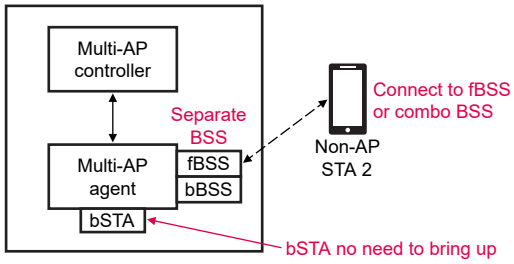

Figure 2. An agent hosted on the local device as the controller

Figure 3 shows the onboarding of an agent. The agent bSTA is used to connect with the comboBSS of an existing agent.

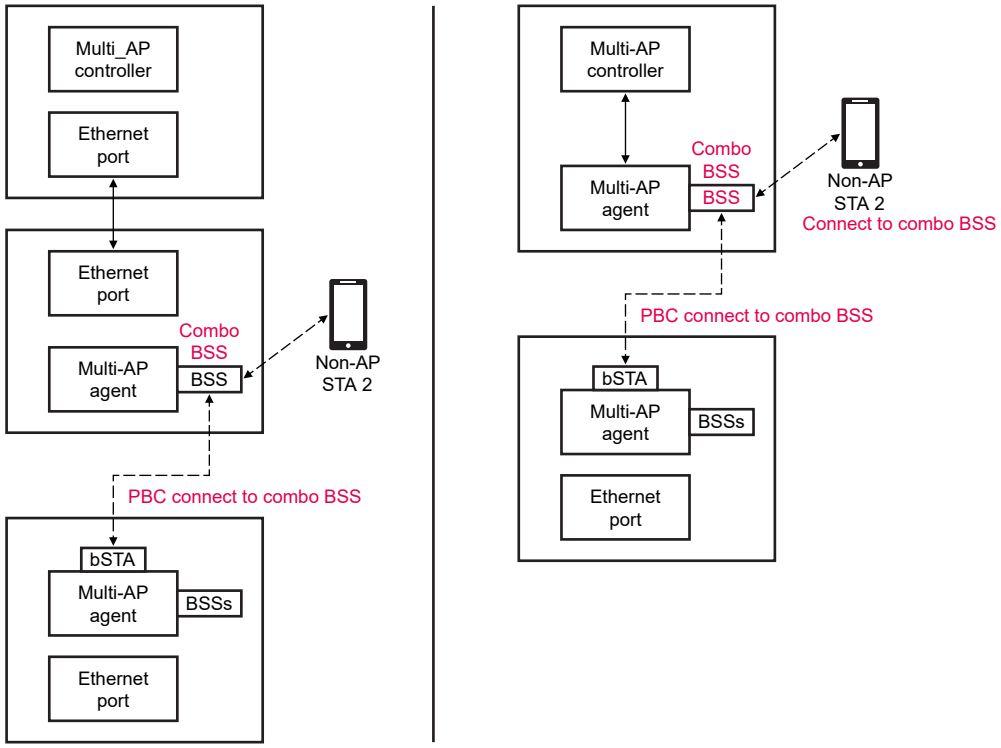

Figure 3. Onboarding of an agent

Figure 4 shows another example of agent onboarding. The agent bSTA is used to connect with the bBSS of an existing agent.

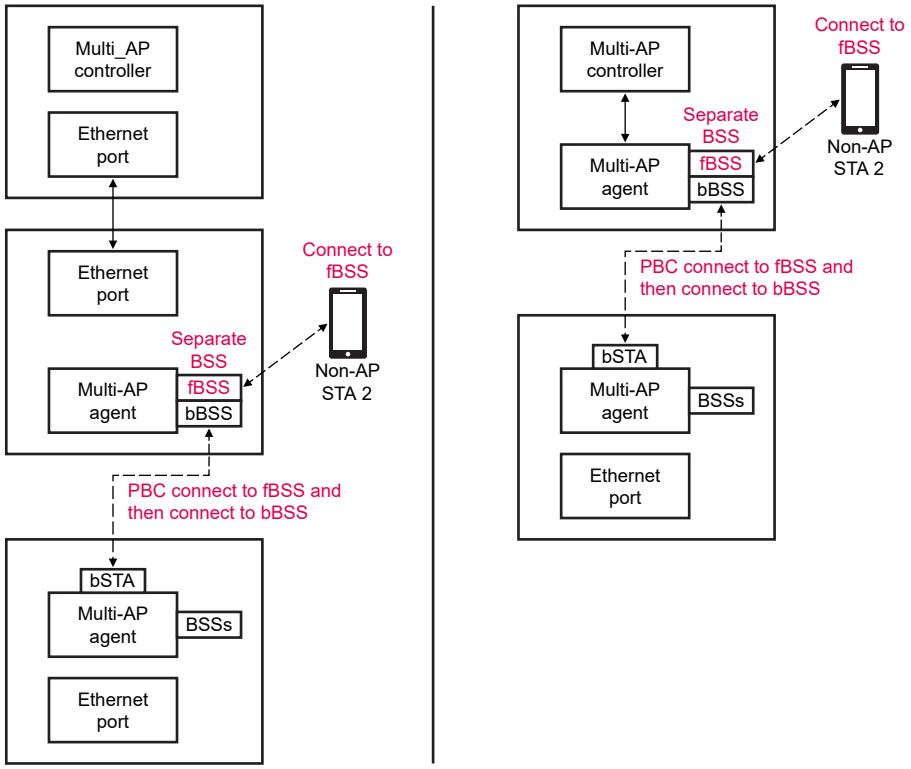

Figure 4. Another example of agent onboarding

Figure 5 shows examples of EasyMesh network formations.

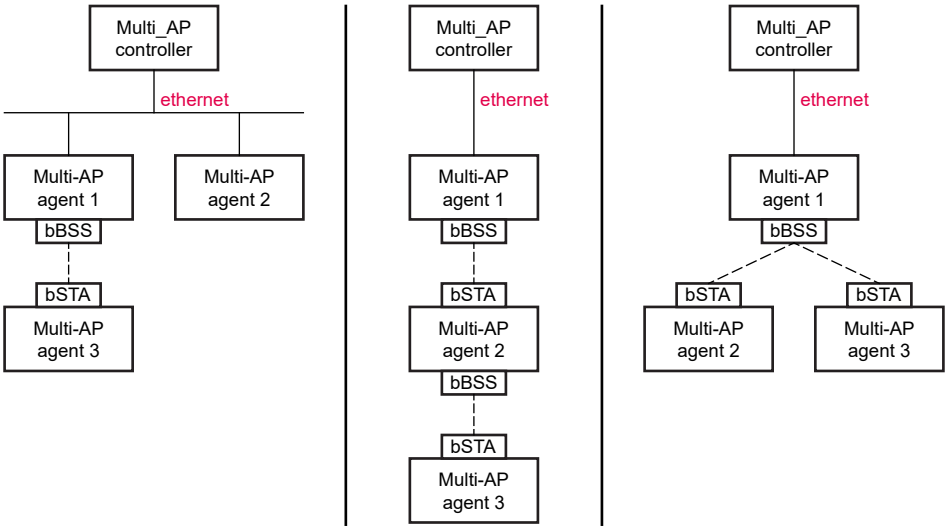

Figure 5. EasyMesh network formations examples

2.4 Onboarding and network formation

Multi-AP onboarding refers to the process of an agent getting the layer-2 connectivity to the Multi-AP network.

For the controller/enrolling agent to onboard a new agent and to form a network, two methods are available:

- Push button configuration (PBC) method:
  - Used with hostapd/wpa_supplicant to onboard an agent by pushing a button.

  - To connect, both the existing controller/agent and the new agent initiate the PBC.
- Device provisioning protocol (DPP) method:
  - A QR code is used to onboard an agent.
  - Enrolling agent creates a QR code in the form of a string. The string is used to onboard a network.
  - Controller must assign agent 1 as the proxy DPP configurator. Agent 1 relays the packets from Air to the controller over Ethernet.
  - Each enrolling agent has a unique QR code that must be shared with the controller.

For the detailed onboarding procedure, refer to Section 6 "Configuration Files".

2.5 Hop count

In the hop count mechanism, each agent assigns itself a hop count value that reflects its index from the controller.

- The hop count (AP index) is included in the beacon frames transmitted by the AP of each agent.
- A hop count value of 1 is assigned to Agent 1.
- Each subsequent agent increases its hop count value by 1 relative to its parent.
- When the bSTA of the agent connects to the bBSS of a parent, the hop count is stored as an internal variable.
- When the BSSs are started on the connecting agent, EasyMesh reads the internal variable and increments the value by 1. The value is added to the vendor-specific IE in beacons.

3 Self-healing

An EasyMesh network has self-healing capabilities. If an intermediate agent becomes non-operational, the neighboring agents automatically scan for available networks and re-establish the connectivity to maintain seamless network performance. To prevent an agent from establishing a loopback connection to a downstream agent, the agent only connects to neighbors with a lower hop count/AP index. This mechanism ensures that the agent reconnects to its parent node direction.

Section 3.1 "Scenario 1" to Section 3.5 "Scenario 5", describe different self-healing scenarios for the multi-AP network.

3.1 Scenario 1

In scenario 1, the multi-AP network is in a serial topology. Agent 2 is powered down. Agent 3 scans the available APs and connects to Agent 1 + Controller as the AP index is lower. Agent 3 does not reconnect to Agent 4 as Agent 4 has a higher AP index.

Figure 6 illustrates scenario 1.

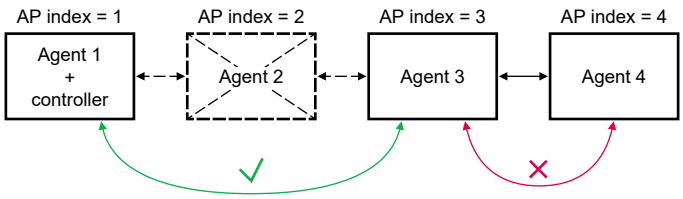

Figure 6. Self-healing scenario 1

3.2 Scenario 2

In scenario 2, Agent 3 is powered down in a serial topology. Agent 4 scans the available APs and can reconnect to either Agent 1 or Agent 2 as both agents have a lower AP index than Agent 4.

Figure 7 illustrates scenario 2.

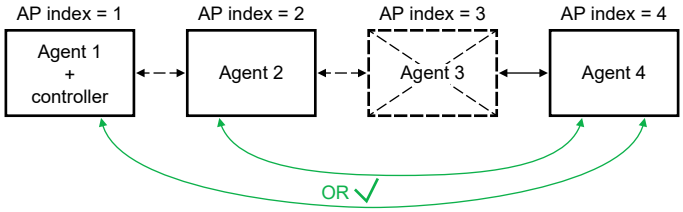

Figure 7. Self-healing scenario 2

3.3 Scenario 3

Figure 8 illustrates scenario 3.

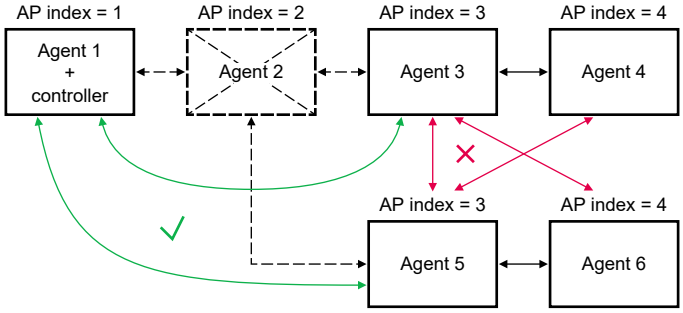

Figure 8. Self-healing scenario 3

3.4 Scenario 4

In scenario 4, Agent 3 is powered down. Agent 4 can either reconnect to Agent 2 or Agent 5 as their AP index is less than 4.

Figure 9 illustrates scenario 4.

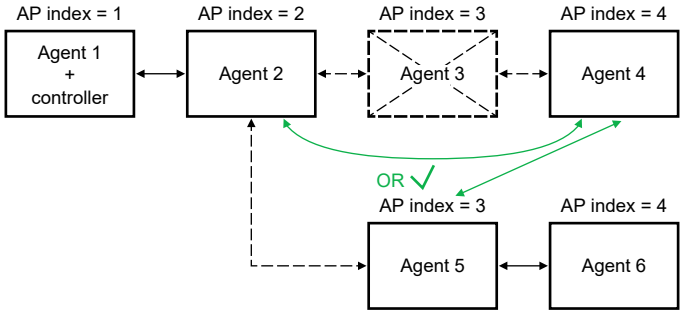

Figure 9. Self-healing scenario 4

3.5 Scenario 5

In scenario 5, Agent 1 + Controller is powered down. Agent 2 and Agent 5 cannot reconnect to each other as their AP indices both equal 2. Instead, Agent 2 and Agent 5 wait for Agent 1 + controller to come back online.

Figure 10 illustrates scenario 5.

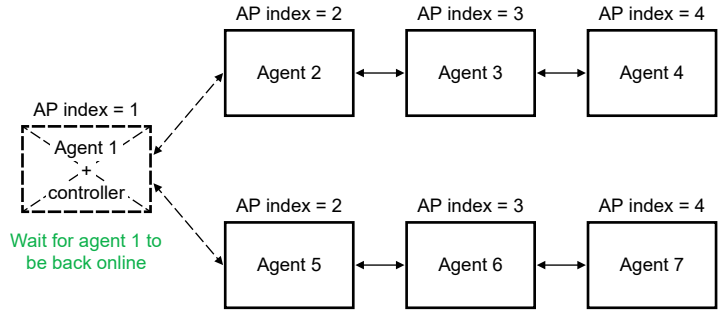

Figure 10. Self-healing scenario 5

4 Software build requirements and Yocto set-up

This section describes how to build a Yocto Linux image with integrated Wi-Fi EasyMesh support, for the FRDM-IMX93 board with the IW612 connectivity module.

4.1 Hardware

- 3x FRDM-IMX93 boards with IW612 Wi-Fi module
  - 1x Controller+Agent device (here onward referred to as "Controller")
  - 2x Standalone Agent devices (here onward referred to as "Agent x")
- 1x Ethernet cable for wired backhaul connectivity
- 1x Ethernet router/switch with Internet connectivity (for WAN uplink)
- Power supplies for all boards

4.2 Host PC Requirements

Host Packages:

Ensure that your host machine is configured with the necessary dependencies for Yocto development. For the complete list of required Ubuntu packages and host setup instructions, refer to the official meta-nxp-connectivity README.

4.3 Download Yocto source code

1. Install the repo tool

```
mkdir -p ~/bin
curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/
repo
chmod a+x ~/bin/repo
export PATH=${PATH}:~/bin
```

2. Initialize and sync the Yocto repository
   Create your workspace and sync the NXP Mickledore release:

```
mkdir ~/imx-yocto-bsp
```

```
cd ~/imx-yocto-bsp
repo init -u https://github.com/nxp-imx/imx-manifest -b imx-linux-mickledore
 -m imx-6.1.55-2.2.0.xml
repo sync
```

3. Check out the NXP Connectivity layer
   Clone the connectivity layer that includes the EasyMesh packages:

```
git clone -b imx_matter_2026_q1 https://github.com/nxp-imx/meta-nxp-
connectivity.git sources/meta-nxp-connectivity
```

Note: *EasyMesh is enabled by default on the imx_matter_2026_q1 branch.*

4.4 Configure and build the Yocto image

Configure and build the Yocto image using any of the following options:

4.4.1 Option A: Build the default matter release (recommended)

To build the standard image for the FRDM-iMX93 with the IW612 Wi-Fi module:

```
cd ~/imx-yocto-bsp
MACHINE=imx93evk-iwxxx-matter DISTRO=fsl-imx-xwayland source sources/meta-nxp-
connectivity/tools/imx-matter-setup.sh bld-xwayland-imx93
bitbake imx-image-multimedia
```

4.4.2 Option B: Add EasyMesh to a standard Yocto BSP

If you are using a regular Yocto Linux BSP and want to add EasyMesh capabilities without integrating the full Matter environment, hook the examples layer in your build setup. Add the following to your bblayers.conf file:

```
BBLAYERS += "${BSPDIR}/sources/meta-nxp-connectivity/meta-nxp-connectivity-
examples"
```

Then, proceed with your standard BitBake build command.

4.5 Flashing the Image

Once the build is complete, flash the resulting `.wic.zst` file to your board. To flash, write directly to a microSD card, or use NXP's Universal Update Utility (UUU).

4.5.1 Method 1: Flashing via SD card (Linux host using dd)

```
# Navigate to the deploy directory
cd ~/imx-yocto-bsp/bld-xwayland-imx93/tmp/deploy/images/imx93evk-iwxxx-matter/
# Decompress the image
zstd -d imx-image-multimedia-imx93evk-iwxxx-matter.rootfs.wic.zst
# Flash to the SD card (Replace /dev/sdX with your actual SD card identifier!)
sudo dd if=imx-image-multimedia-imx93evk-iwxxx-matter.rootfs.wic of=/dev/sdX
 bs=4M conv=fsync status=progress
```

4.5.2 Method 2: Flashing via USB (using NXP's UUU tool)

NXP’s UUU tool allows you to flash the board directly over a USB-C connection using Serial Download Mode.

1. Set the board's boot switches to Serial Download Mode.
2. Connect the board's USB1_C port to your host PC.
3. Run the following command (requires UUU to be installed on your host):

```
sudo uuu -b emmc_all imx-image-multimedia-imx93evk-iwxxx-
matter.rootfs.wic.zst
```

Note: *After flashing, remember to change the boot switches back to SD card/eMMC boot mode before powering on.*

4.6 First boot configuration (FRDM-IMX93 specific)

Important: *The FRDM-IMX93 requires a one-time device tree configuration in U-Boot.*

1. Insert the flashed microSD card into the board and power it on.
2. During the boot sequence, press any key to halt autoboot and enter the U-Boot console.
3. Configure the correct device tree file for the FRDM board:

```
u-boot=> setenv fdtfile imx93-11x11-frdm.dtb
u-boot=> saveenv
u-boot=> boot
```

Repeat this setup for all three boards (one controller, two agents) in your mesh network.

4.7 Verify EasyMesh software installation

After the device boots to Linux, log in as root and verify that the EasyMesh binaries are present:

```
# Check for EasyMesh executables
which easymesh-controller
which easymesh-agent
which easymesh_cli
```

5 Network bring-up

The physical Multi-AP setup is constructed dynamically based on your operational requirements. The initial onboarding connects agent 1 over a physical Ethernet cable to the Multi-AP controller. Then, other agents, such as agent 2, can be wirelessly provisioned into the established mesh configuration.

This section covers two major onboarding categories:

1. Legacy method (1905.1 auto-discovery or Wi-Fi push button configuration)
2. Enhanced DPP method (Device Provisioning Protocol using QR code URIs)

5.1 Controller initialization

The controller represents the absolute center of the network topology. Its primary functions are directing backhaul and fronthaul interfaces, aggregating network capability data, onboarding more agents, and serving as the primary bridge network host.

5.1.1 Preparing the system environment

Ensure that you have completed all hardware setup guidelines outlined in Section 4 "Software build requirements and Yocto set-up" and that the evaluation board is booted to a command prompt.

Step 1: Validate the hardware interfaces and install necessary configuration file parameters (refer to Section 6.1 "wlan_mc.config (Controller)" and Section 6.3 "wifi_mod_para.conf (Driver parameters)"). Adjust the max_uap_bss value to support simultaneous UAP functions.

Step 2: Ensure that module dependencies are refreshed prior to initialization:

```
depmod -a
```

Step 3: Begin the background system service to start the controller:

```
systemctl start easymesh-controller
```

Step 4: Assign the IP address to br0 as shown in Section 5.1.1.1 "Option A: Use a built-in DHCP server (Standalone network)" or Section 5.1.1.2 "Option B: Use an existing DHCP server (Shared network)" with updated index number.

Step 5: Configure the DHCP service.

5.1.1.1 Option A: Use a built-in DHCP server (Standalone network)

If your controller is the primary network gateway (no existing DHCP server on the Ethernet backhaul), create a lightweight DHCP service:

```
cat << 'EOF' > /tmp/udhcpd.conf
start 192.168.10.100
end 192.168.10.200
interface br0
opt router 192.168.10.1
opt dns 8.8.8.8
opt subnet 255.255.255.0
EOF
udhcpd /tmp/udhcpd.conf
```

5.1.1.2 Option B: Use an existing DHCP server (Shared network)

If an external DHCP server exists on your Ethernet backhaul:

1. Skip the `udhcpd` setup above.
2. Configure the controller's `br0` interface to obtain an IP via DHCP:

   ```
   udhcpc -i br0
   ```

3. Ensure that the external DHCP server has a sufficient IP address pool for accommodating all agents and client devices.
4. Configure routing on the external router to allow Internet access through the EasyMesh network.

Note: *For production deployments with Internet access requirements, Option B with an existing router/DHCP server is recommended.*

5.2 Agent 1 onboarding via Ethernet

In this architecture, agent 1 physically connects to the controller. Due to this wired configuration, an active Wi-Fi backhaul (bSTA) link is not used. Instead, the Multi-AP backhaul routes strictly across the eth0 interface.

5.2.1 Legacy method for agent 1 onboarding over Ethernet

The legacy method relies on zero-configuration mechanisms (via IEEE 1905.1).

Step 1: Establish the physical Ethernet link between the controller's eth interface and agent 1's eth interface.

Step 2: Disable IPv6 functionality locally:

```
sysctl -w net.ipv6.conf.all.disable_ipv6=1
sysctl -w net.ipv6.conf.default.disable_ipv6=1
```

Step 3: Apply the necessary driver components:

```
modprobe mlan
modprobe moal mod_para=nxp/wifi_mod_para.conf
```

Step 4: Launch the Multi-AP agent systemd service:

```
systemctl start easymesh-agent
```

Step 5: Force the local bridge (`br0`) to solicit an IP lease from the controller's active DHCP host:

```
udhcpc -i br0
```

At this point, agent 1 autonomously maps itself directly to the controller's topology.

5.2.2 Enhanced DPP method for agent 1 onboarding over Ethernet

This method establishes strict network security parameters using cryptographic Universal Resource Identifier (URI) configurations (known as a "QR code"). It establishes Agent 1 as a trusted "Configurator Proxy", which is fundamentally required to add wireless agents via DPP later.

Note: *Ensure that you follow Section 5.2.1 (Steps 1–4) before proceeding.*

Step 1 (Agent 1): To construct the URI Hash based on local hardware MAC address values, execute the native EasyMesh command interface.

```
easymesh_cli -c get_dpp_bootstrap_uri type=qrcode
```

Output Example:

```
DPP:V:2;K:MDkwEwYHKOZIzj0CAQYIKOZIzj0DAQcDIgACSxc...
```

Step 2 (Controller): Instruct the active Controller daemon to accept this precise URI that helps securely adds Agent 1 to the authorized peer database.

Note: *Use double quotes around the string.*

```
easymesh_cli -c add_dpp_bootstrap_uri
"DPP:V:2;K:MDkwEwYHKOZIzj0CAQYIKOZIzj0DAQcDIgACSxc..."
```

Step 3 (Agent 1): Initiate a DPP discovery sequence (Chirp) designated on the ethx port.

```
easymesh_cli -c start_dpp_chirp eth0
```

Step 4 (Controller): Assign the newly authenticated Agent 1 as the designated proxy DPP configurator. This enables Agent 1 to listen for future wireless connections on behalf of the wired Controller.

```
easymesh_cli -c set_dpp_proxy_agent
```

5.3 Agent 2 onboarding via Wi-Fi

Expanding the mesh topology without the constraints of physical Ethernet lines requires onboarding an Agent via wireless Multi-AP backhaul capabilities. Here, Agent 2 operates with both an active fronthaul AP (fBSS) and a backhaul STA (bSTA).

5.3.1 PBC method for Agent 2 and beyond onboarding over Wi-Fi

The legacy wireless methodology uses Wi-Fi Protected Setup Push-Button Configuration (WPS-PBC) to expose a pairing window temporarily.

Step 1: Establish Agent 2's base state environment.

```
modprobe mlan
modprobe moal mod_para=nxp/wifi_mod_para.conf
sysctl -w net.ipv6.conf.all.disable_ipv6=1
sysctl -w net.ipv6.conf.default.disable_ipv6=1
```

Step 2 (Agent 2): Launch the Multi-AP agent service.

```
systemctl start easymesh-agent
```

Step 3 (Agent 2): Initiate the enrollee WPS pairing sequence via wpa_supplicant on the Multi-AP interface.

```
wpa_cli wps_pbc multi_ap=1
```

Step 4 (Agent 1): To begin listening for Agent 2's connection request actively on its UAP interface, initiate the configurator pairing sequence.

```
hostapd_cli -i uap0 wps_pbc
```

Step 5 (Agent 2): Allow 10-15 seconds for connection, verify mlan0 state, and then solicit an IP lease over the newly established wireless link.

```
iw dev mlan0 link udhcpc -i br0
```

5.3.2 Enhanced DPP method for Agent 2 and beyond onboarding

This method is the most secure method for wirelessly onboarding devices. It operates independent of wpa_cli execution, using native backend daemons.

Prerequisite: The Controller must have assigned Agent 1 as a proxy (refer to Step 4 in Section 5.2.2 "Enhanced DPP method for agent 1 onboarding over Ethernet"). Otherwise, the Controller cannot process the wireless chirp.

Step 1: Follow Section 5.3.1 Steps 1 and 2 to initialize Agent 2.

Step 2 (Agent 2): Generate the unique URI for this specific node:

```
easymesh_cli -c get_dpp_bootstrap_uri type=qrcode
```

Step 3 (Agent 2): Transmit out the generated configuration request over the primary Wi-Fi antenna interface:

```
easymesh_cli -c start_dpp_chirp mlan0
```

Step 4 (Controller): Instruct the Controller to authenticate the specific string actively generated in Step 2. Agent 1 (operating as the proxy) relays this validation:

```
easymesh_cli -c add_dpp_bootstrap_uri
"DPP:V:2;K:MDkwEwYHKOZIzj0CAQYIKOZIzj0DAQcDIgACHSG..."
```

Step 5 (Agent 2): Verification and address assignment.

Native background services handle passing the secure Wi-Fi connection elements. Wait 10 seconds, and then request an IP.

```
iw dev mlan0 link
udhcpc -i br0
```

6 Configuration Files

Specific configuration files located in the `/etc/easymesh/` directory, drives the EasyMesh behavior.

6.1 wlan_mc.config (Controller)

The file is used to configure the controller with the BSS information parameter `bss_info`. The `bss_info1` corresponds to bBSS. The `bss_info2` corresponds to fBSS.

Example of wlan_mc config content:

```
#### User configuration
driver_name = W8XXX
manufacturer_name = NXP
model_name = WIFI_PHY_x200
model_number = 12345
serial_number = 100000000000
device_name = NXP_EasyMesh_Controller
control_url = http://192.168.10.1
uuid = 100000000000
interface_type = INTERFACE_TYPE_IEEE_802_3AB_GIGABIT_ETHERNET
#### Runtime automatic configuration
power_state = INTERFACE_POWER_STATE_ON
#### When controller is running as a wireless device (STA)
# ieee80211.bssid = 00:00:00:00:00:00
# ieee80211.role = IEEE80211_ROLE_AP
# ieee80211.ap_channel_width = 3
# ieee80211.ap_channel_center_frequency_index_1 = 00
# ieee80211.ap_channel_center_frequency_index_2 = 00
#### BSS Info
bss_info1=FF:FF:FF:FF:FF:FF 11x EasyMesh-5G 0x0040 0x0008 maprocks1 1 1
#bss_info1=FF:FF:FF:FF:FF:FF 11x EasyMesh-5G_BH 0x0040 0x0008 maprocks1 1 0
#bss_info2=FF:FF:FF:FF:FF:FF 11x EasyMesh-5G_FH 0x0040 0x0008 maprocks1 0 1
```

The lines with `bss_info` or `comboBSS` in `wlan_mc.config` include the editable/configurable parameters:

- `bss_info1` corresponds to bBSS.
- `bss_info2` corresponds to fBSS.
- For `comboBSS`, set the last 2 arguments to 1 1.

Syntax for `bss_info`:

```
bss_info<N>= <temp_BSSID> <band> <SSID> <Auth_type> <Encryption_mode>
<password><Backhaul><Fronthaul>
```

Table 1. Command parameters

Parameter	Description
N	BSSID configurations for each access point of topology for set-up N = BSS information number, starts at 1
temp_BSSID	Hardware address Set to ff:ff:ff:ff:ff:ff as it is unknown
band	Band mode 8x = 2.4 GHz 11x = 5 GHz
SSID	Service set identifier (SSID) Ensure it matches with the hostapd SSID
Auth_type	Authorization type 0x0008 = IEEE80211_AUTH_MODE_WPA 0x0010 = IEEE80211_AUTH_MODE_WPA2 0x0020 = IEEE80211_AUTH_MODE_WPA2PSK 0x0040 = IEEE80211_AUTH_MODE_SAE
Encryption_mode	Encryption modes 0x0001 = WPS_ENCR_NONE 0x0004 = WPS_ENCR_TKIP 0x0008 = WPS_ENCR_AES
Password	Network key/Password used in all applications
Backhaul	Disable/Enable backhaul BSS 0 = disable 1 = enable Set to 1
Fronthaul	Disable/Enable fronthaul BSS 0 = disable 1 = enable Set to 1

6.2 wlan_ma.config (Agent)

The file is used for the configuration of the EasyMesh agent. It contains default parameters, such as `driver_name` and `model_name`. It is highly recommended to align the `ieee80211.ap_channel_default` parameter across all nodes to ensure that the backhaul networks discover each other smoothly.

```
#### User configuration
driver_name = W8XXX
manufacturer_name = NXP
```

```
model_name = WIFI_PHY_x200
model_number = 12345
serial_number = 5ce6c9
device_name = NXP_EasyMesh_Agent
control_url = http://192.168.10.44
uuid = 075ECB0B-6129-7C29-FA5B-CC6F40100899
interface_type = INTERFACE_TYPE_IEEE_802_11AX_5_GHZ
ethernet_backhaul_port = eth0
bss.name = uap
ieee80211.max_bss = 1
ieee80211.country_code = US
ieee80211.ap_channel_default = 36
# ap_channel_width: 0=20MHz 1=40MHz 2=80MHz 3=160MHz
ieee80211.ap_channel_width = 3
ieee80211.role = IEEE80211_ROLE_AP
#### Runtime automatic configuration
ieee80211.bssid = 00:00:00:00:00:00
ieee80211.bssid_5G = 00:00:00:00:00:00
ieee80211.ssid = EasyMesh-5G
ieee80211.authentication_mode = IEEE80211_AUTH_MODE_WPA2PSK
ieee80211.encryption_mode = IEEE80211_ENCRYPTION_MODE_AES
ieee80211.network_key = maprocks1
#ieee80211.complete_config_radio0 = 0
#ieee80211.complete_config_radio1 = 0
#ieee80211.complete_config_radio2 = 0
power_state = INTERFACE_POWER_STATE_ON
radio0.ruid = 00:00:00:00:00:00
radio0.policy = FF
radio0.channel_util_threshold = 00
radio0.rssi_steering_threshold = 00
radio0.metrics_policy = 00
radio1.ruid = 00:00:00:00:00:00
radio1.policy = FF
radio1.channel_util_threshold = 00
radio1.rssi_steering_threshold = 00
radio1.metrics_policy = 00
local_steering_disallowed1 = 00:00:00:00:00:00
local_steering_disallowed2 = 00:00:00:00:00:00
local_steering_disallowed3 = 00:00:00:00:00:00
local_steering_disallowed4 = 00:00:00:00:00:00
local_steering_disallowed5 = 00:00:00:00:00:00
btm_steering_disallowed1 = 00:00:00:00:00:00
btm_steering_disallowed2 = 00:00:00:00:00:00
btm_steering_disallowed3 = 00:00:00:00:00:00
btm_steering_disallowed4 = 00:00:00:00:00:00
btm_steering_disallowed5 = 00:00:00:00:00:00
#### Default target bssid for steering request
target_bssid = FF:FF:FF:FF:FF:FF
```

Use the configuration file to change the channel, the band, and the combo BSS setting. Table 2 lists the frequently modified parameters.

Table 2. Frequently modified parameters in the wlan_ma.config file

Parameter	Description
ieee80211.ap_channel_default	AP channel
interface_type	Interface type INTERFACE_TYPE_IEEE_802_11B_2_4_GHZ INTERFACE_TYPE_IEEE_802_11G_2_4_GHZ INTERFACE_TYPE_IEEE_802_11A_5_GHZ INTERFACE_TYPE_IEEE_802_11N_2_4_GHZ INTERFACE_TYPE_

Table 2. Frequently modified parameters in the wlan_ma.config file*...continued*

Parameter	Description
	IEEE_802_11N_5_GHZ INTERFACE_TYPE_IEEE_802_11AC_5_GHZ INTERFACE_TYPE_IEEE_802_11AX_5_GHZ INTERFACE_TYPE_IEEE_802_11AX_2_4_GHZ
`ieee80211.max_bss`	Used by the agent to notify the controller that combo or separated fronthaul and backhaul BSS is supported. 1 = combo BSS 2 = separate BSS

6.3 wifi_mod_para.conf (Driver parameters)

Before loading the Wi-Fi drivers, the module parameter file must be configured to support multiple virtual BSSs (required for simultaneous fronthaul and backhaul APs).

Path: `/lib/firmware/nxp/wifi_mod_para.conf`

Modification: Set max_uap_bss = 2

This parameter sets the maximum UAP BSS, allowing the agent to broadcast both fronthaul and backhaul networks concurrently.

For example, to set for IW612:

```
SDIW612 = {
cal_data_cfg=none
ext_scan=1
max_uap_bss=2
fw_name=nxp/sduart_nw61x_v1.bin.se
}
```

6.4 Advanced configuration: Separate fronthaul and backhaul BSS

By default, EasyMesh agents use a Combo BSS configuration where a single BSS handles both fronthaul (client connections) and backhaul (mesh connectivity).

For advanced deployments requiring separate BSS for fronthaul and backhaul, follow these configuration steps:

Step 1: Modify driver parameters.

Edit `/lib/firmware/nxp/wifi_mod_para.conf` and set:

```
max_uap_bss=2
```

Step 2: Update `wlan_ma.config`.

Set the following parameter:

```
ieee80211.max_bss = 2
```

Step 3: Update `wlan_mc.config`.

Configure separate BSS information:

```
bss_info1=FF:FF:FF:FF:FF:FF 11x EasyMesh-5G_BH 0x0040 0x0008 maprocks1 1 0 #
 Backhaul only
```

```
bss_info2=FF:FF:FF:FF:FF:FF 11x EasyMesh-5G_FH 0x0040 0x0008 maprocks1 0 1 #
 Fronthaul only
```

Note: *This configuration is optional and recommended only for advanced network topologies requiring isolated fronthaul and backhaul traffic.*

7 Runtime configuration commands

Once the EasyMesh network is set up, use the commands in this section to alter the configuration in runtime. The command-line interface (CLI) for the easymesh application natively in the path is `easymesh_cli`.

During runtime, `easymesh_cli` can be used to:

- Trigger network topology mapping.
- Send channel preferences to an agent.
- Steer the client.

7.1 Network topology mapping

Once the network is established, the Controller can generate a full mapping of all connected fronthaul and backhaul nodes and their MAC addresses.

Step 1: To trigger the topology map, run the following command on the Controller:

```
easymesh_cli -t 0xFF00
```

Step 2: The native service processes this request in the background. Read the generated JSON by checking the tail of the log file:

```
tail -n 50 /tmp/easymesh-controller.log
```

7.2 Channel preference and channel selection

Command syntax:

```
easymesh_cli -m <AL MAC of Agent> -t 0x8004
easymesh_cli -m <AL MAC of Agent> -t 0x8006
```

Where:

- `-m <AL_MAC>`: Target agent's abstraction layer MAC address
- `-t <MSG_TYPE>`: 1905.1 message type code
- Message type codes:
  - `0x8004`: Channel preference query (requests agent's preferred channels)
  - `0x8006`: Channel selection request (instructs agent to change channel)

Step 1: Send the channel preference query to the agent:

```
easymesh_cli -m <AL MAC of Agent> -t 0x8004
```

The agent sends a channel preference report to the controller with a preference value assigned for each operating class.

Step 2: Send a channel selection request:

```
easymesh_cli -m <AL MAC of Agent> -t 0x8006
```

The controller sends a channel selection request to the agent by choosing the highest preference operating classes, from the channel preference report of the agent. The agent selects the highest preference operating class with highest bandwidth, and switches to the first channel available in the channel list of operating classes.

7.3 Client steering

Client steering enables the network to guide devices to the optimal access point or frequency band for better performance. The feature enhances the connectivity by balancing the load and improving the roaming decisions across the mesh.

7.3.1 Configure steering policy in agents

Configure the steering policy using an `easymesh_cli` command. The command sets the agent-side steering behavior, including disallowed STA lists, per-radio steering policies, and thresholds for channel utilization and received channel power indicator (RCPI).

Command syntax:

```
easymesh_cli -t 8003 -m <AL_MAC> -j <continuous_hex_payload>
```

Note: *The -j payload parameter must be formatted as a continuous hex string without spaces or colons.*

Payload structure (in order):

```
<Local steering Disallowed STA count><STA MACs><BTM steering disallowed
 count><STA MACs><Radio count><Radio MAC><steering policy><Utilization
 threshold><RCPI threshold>
```

Example:

Configure Agent `a0:cd:f3:77:e6:1c` to allow steering on a specific radio with defined thresholds:

```
easymesh_cli -t 8003 -m a0:cd:f3:77:e6:1c -j 000001a2cdf377e71c00ffb4
```

Parameter breakdown (for the hex payload 000001a2cdf377e71c00ffb4):

- `00`: Local steering Disallowed STA count (0 STAs)
- `00`: BTM steering disallowed count (0 STAs)
- `01`: Radio count (1 Radio to configure)
- `a2cdf377e71c`: Radio MAC address
- `00`: Steering policy (All steering allowed)
- `ff`: Channel utilization threshold
- `b4`: RCPI threshold

7.3.2 Controller initiated client steering request

Configure the steering request using an `easymesh_cli` command. The command instructs the agent to steer a specific client (STA) to a target BSSID (AP), under defined conditions such as target AP, channel number, operating class.

Command syntax:

```
easymesh_cli -t 8014 -m <AL_MAC> -j <continuous_hex_payload>
```

Payload structure (in order):

```
<Source BSSID><steering req mode><Steering Opportunity window><BTM Disassoc
 Timer><STA count><STA MAC><Target BSSID count><Target BSSID><Oper class><Chan
 number>
```

Example:

Steer STA `6c:c7:ec:90:5f:e5` from source BSSID `a2:cd:f3:77:e7:1c` to the target BSSID `22:ba:36:5c:e7:c9` (OperClass 116, Channel 36):

```
easymesh_cli -t 8014 -m a0:cd:f3:77:e6:1c -j
 a2cdf377e71ce000000138016cc7ec905fe50122ba365ce7c97424
```

Parameter breakdown (for the hex payload a2cdf377e71ce000000138016cc7ec905fe50122ba365ce7c97424):

- `a2cdf377e71c`: Source BSSID (Where the client is connected currently)
- `e0`: Steering mode (Request mode, not mandate)
- `0000`: Steering opportunity window
- `0138`: BTM disassociation timer
- `01`: STA count (Number of STAs to steer)
- `6cc7ec905fe5`: STA MAC address to steer
- `01`: Target BSSID count (Number of target APs)
- `22ba365ce7c9`: Target BSSID (Destination AP)
- `74`: Operating class (Hex equivalent for OperClass 116)
- `24`: Channel number (Hex equivalent for Channel 36)

Verification:

Verify that the STA is steered to the targeted BSS as per the steering policy and request by executing the following on the Agent:

```
hostapd_cli -i uap0 all_sta
```

8 Performance benchmark

An EasyMesh network supports up to 10 agents within a single deployment. Table 3 demonstrates the expected network performance—specifically throughput (Mbit/s) and ping latency (msec)—as the mesh topology scales to its maximum depth. These measurements were obtained using the IW612 module operating on the 5 GHz band with 80 MHz channel bandwidth. As the hop count increases from Agent 1 to Agent 10, these results illustrate the natural degradation of bandwidth and the corresponding increase in latency across the backhaul links.

Test configuration:

- Device: IW612 (5 GHz, 80 MHz bandwidth)
- Traffic: TX UDP (from Agents to Controller)
- Environment: Shielded chamber, channel 149

Table 3. Multi-AP scaling performance metrics

Traffic to backend server		Ping latency		
		Min	Avg	Max
Agent 1	N/A	N/A	N/A	N/A
Agent 2	325 Mbit/s	3.837	4.137	4.402
Agent 3	135 Mbit/s	5.723	6.341	6.633
Agent 4	85 Mbit/s	8.361	9.236	9.816
Agent 5	60 Mbit/s	11.269	11.893	13.351
Agent 6	50 Mbit/s	12.929	13.788	15.009
Agent 7	40 Mbit/s	15.959	16.338	16.627
Agent 8	25 Mbit/s	17.711	18.933	20.639
Agent 9	22 Mbit/s	19.931	20.85	22.621
Agent 10	18 Mbit/s	21.711	23.264	26.922

Note: *Agent 1 hosts Controller and Agent services, therefore, performance data is not applicable. Agent 2 to Agent 10 show Wi-Fi mesh backhaul performance using IW612 (5 GHz, 80 MHz bandwidth).*

9 References

[1] Webpage – 88Q9098: 2.4/5 GHz Dual-band 2x2 Wi-Fi 6 (802.11ax) + Bluetooth Automotive Solution (link)
[2] Webpage – 88W9098: 2.4/5 GHz Dual-band 2x2 Wi-Fi 6 (802.11ax) + Bluetooth (link)
[3] Webpage – AW611: 2.4/5 GHz Dual-band 1x1 Wi-Fi 6 (802.11ax) + Bluetooth Automotive Solution (link)
[4] Webpage – IW611: 2.4/5 GHz Dual-band 1x1 Wi-Fi 6 (802.11ax) + Bluetooth Solution (link)
[5] Webpage – IW612: 2.4/5 GHz Dual-band 1x1 Wi-Fi 6 (802.11ax) + Bluetooth + 802.15.4 Tri-radio Solution (link)
[6] Webpage – IW610: 2.4/5 GHz Dual-Band 1x1 Wi-Fi 6 + Bluetooth Low Energy 5.4 + 802.15.4 Tri-Radio Solution (link)

10 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

11 Revision history

Table 4 summarizes the revisions to this document.

Table 4. Revision history

Document ID	Release date	Description
AN14996 v.1.0	28 April 2026	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

Matter, Zigbee — are developed by the Connectivity Standards Alliance. The Alliance's Brands and all goodwill associated therewith, are the exclusive property of the Alliance.

Tables

Figures

Contents