periMICA

Containerized Wi-Fi® AP and Wi-Fi Station Setup



Application Note

Perinet
Seamless IoT Connectivity

Abstract

A *periMICA* can be a Wi-Fi® AP of its own, using a USB adapter that is driven entirely from a container. This Application Note describes how to set this up with the periMICA Debian container, from the driver prerequisites of the adapter through handing the wireless interface to the container to the running Wi-Fi AP. It also describes how the same adapter makes the periMICA a Wi-Fi Station of an existing wireless network, and how the wired devices can be routed over that wireless uplink.

Doc-No.: PRN.100.991 rev: 1 2026-11-04

Document Information

Title	periMICA
Subtitle	Containerized Wi-Fi® AP and Wi-Fi Station Setup
Type	Application Note
Status	Release
Version	1
Date	2026-11-04
Disclosure Restriction	

periCORE and all periCORE based products, such as Perinet Smart Components, are subject to property rights from various patents, e.g. [4, 5, 3]

Further intellectual property rights in the products, names, logos and designs included in this document may be held by *Perinet* or third parties. Copying, reproduction, modification or disclosure to third parties of this document or any part thereof is only permitted with the express written permission of *Perinet*.

The information contained herein is provided “as is” and *Perinet* assumes no liability for its use. No warranty, either express or implied, is given, including but not limited to, with respect to the accuracy, correctness, reliability and fitness for a particular purpose of the information. This document may be revised by *Perinet* at any time without notice. For the most recent documents, visit https://perinet.io.

Copyright © Perinet GmbH.

Contents

1 Introduction 5
- 1.1 Scope . . . 5
- 1.2 Prerequisites . . . 5

2 Adapter Prerequisites 7
- 2.1 Tested adapter . . . 7
- 2.2 Drivers in the periMICA base system . . . 7
- 2.3 Checking an adapter . . . 8
- 2.4 Frequency bands . . . 9

3 Preparing the Debian Container 10
- 3.1 Installing the packages . . . 10
- 3.2 Keeping the container self-contained . . . 10

4 Handing the Radio to the Container 12
- 4.1 Moving the radio . . . 12

5 Running the Wi-Fi AP 13
- 5.1 Regulatory domain . . . 13
- 5.2 Configuring the Wi-Fi AP . . . 13
- 5.3 Broadcast and multicast with rtw drivers . . . 14
- 5.4 Starting and verifying . . . 15
- 5.5 Starting automatically . . . 15
- 5.6 Bridged . . . 16
- 5.7 Routed . . . 17
- 5.8 Without address configuration . . . 17

6 Running as a Wi-Fi Station 18
- 6.1 Adapter requirements . . . 18
- 6.2 Installing the packages . . . 18
- 6.3 Configuring the connection . . . 18
- 6.4 Connecting and obtaining addresses . . . 19
- 6.5 Starting automatically . . . 20
- 6.6 Routing the wired network over the wireless uplink . . . 20

7 Troubleshooting 23

8 Contact & Support 25

A List of Figures 26

B List of Tables 27

C Glossary 28

D References 29

E List of Patents 30

F Revision History 31

1 Introduction

A periMICA has no radio of its own. Wireless connectivity is added with a USB Wi-Fi® adapter, and everything that drives that adapter can live inside a container. This Application Note describes how to turn a periMICA into a Wi-Fi® access point using the periMICA Debian container, so that notebooks, tablets or handheld terminals associate directly with the periMICA.

Two roles are distinguished throughout this document:

- **Wi-Fi AP**: the device that operates a Wi-Fi network, which other devices associate with. The periMICA in this role is the subject of Section 2 to Section 5.
- **Wi-Fi Station**: a device that associates with a Wi-Fi AP and uses its network. The notebooks, tablets and terminals above are Wi-Fi Stations, and the periMICA becomes one itself when it joins an existing network (Section 6).

Perinet provides a ready-made container that makes the periMICA a Wi-Fi AP, the *Wi-Fi Bridge* container [2]. It runs the Wi-Fi AP on the adapter handed to it, bridges the Wi-Fi Stations with the network of the periMICA, and is set up through its web interface: band, channel and country are chosen automatically, and WPA2/WPA3 Personal protects the network. This Application Note describes the manual setup in the Debian container, for a Wi-Fi AP of one's own design or to understand what the Wi-Fi Bridge container does.

Running the Wi-Fi AP from a container rather than from the base firmware has three practical consequences:

- The base system stays untouched. It provides the driver and hands the wireless interface over, but needs no wireless tooling of its own.
- Radio configuration and the Wi-Fi AP daemon are delivered, versioned and replaced together with the container that uses them.
- A firmware update of the periMICA does not overwrite the configuration, because the container carries it in its own persistent storage.

1.1 Scope

This document covers the complete path from a freshly installed Debian container to a Wi-Fi AP that beacons and accepts Wi-Fi Stations: driver prerequisites, handing the wireless interface to the container, and the Wi-Fi AP daemon itself. Integrating its Wi-Fi Stations into the wired network is outlined in Section 5.6. Connecting the periMICA to an existing wireless network as a Wi-Fi Station, and routing the wired devices over that connection, is described in Section 6.

1.2 Prerequisites

- periMICA firmware 24-05 or later. The drivers for the wireless adapters covered here are part of that release, see Section 2.

- The Debian container, installed and started. Container installation, mDNS naming and SSH access are described in the *Debian Container User Guide* [1].
- A USB Wi-Fi® adapter whose driver supports Wi-Fi AP operation.
- A workstation with SSH access to the container.

Note: The adapter driver has to be part of the periMICA Linux kernel, and it is the kernel that decides which adapters can be used at all. Support for the adapters described in this document is provided by periMICA firmware 24-05 (not yet released). On earlier firmware versions such an adapter is not recognised, no matter how the container is configured.

2 Adapter Prerequisites

Not every USB Wi-Fi® adapter can act as a Wi-Fi AP. Two conditions have to be met: the driver for the adapter has to be part of the periMICA firmware, which is the case from release 24-05 onwards for the adapters listed below, and that driver has to support Wi-Fi AP operation. The second condition is the one that is easy to overlook. A driver reports to the kernel which operating modes its chip supports, and adapters sold as "client adapters" frequently report Wi-Fi Station mode only. Such an adapter scans and associates perfectly well, but the Wi-Fi AP daemon cannot start on it, and no amount of configuration changes that.

2.1 Tested adapter

The setup described in this document was validated with a TP-Link Archer T2UB Nano USB adapter. It is based on the Realtek RTL8821CU, a single-stream 802.11ac combination chip that also carries Bluetooth:

Adapter	TP-Link Archer T2UB Nano
Chip	Realtek RTL8821CU
USB ID	0bda:c820, reported as "802.11ac NIC"
Driver	rtw_8821cu
Firmware	rtw88/rtw8821c_fw.bin
Validated	2.4 GHz with HT20, and 5 GHz with 80 MHz channel width
Known issue	Loses its Wi-Fi Stations under the broadcast and multicast traffic of a busy network, see Section 5.3; the settings there were validated with an RTL8812AU adapter

Table 1: The adapter used for this Application Note

An RTL8812AU adapter (USB ID 0bda:8812, driver rtw_8812au) was tested as a Wi-Fi AP as well, on 5 GHz with 80 MHz channel width and the settings of Section 5.3. Under sustained traffic it ran without a driver error for the duration of the test.

2.2 Drivers in the periMICA base system

The periMICA kernel contains the drivers below for adapters on USB. Whether an adapter works as a Wi-Fi AP is a property of its driver:

Driver	Wi-Fi AP	Chips
rtw_8821cu	yes, tested	RTL8821CU
rtw_8812au	yes	RTL8812AU
rtw_8814au	yes	RTL8814AU
rtw_8821au	yes	RTL8821AU
rtw_8822bu	yes	RTL8822BU
rtw_8822cu	yes	RTL8822CU
rtw_8723du	yes	RTL8723DU
rtl8192cu	yes	RTL8188CU, RTL8192CU
rt2800usb	yes	Ralink RT2770 to RT5572
rtl8xxxu	**no**	RTL8188, RTL8192, RTL8723
mt7601u	**no**	MediaTek MT7601U

Table 2: USB wireless drivers and their Wi-Fi AP support

The seven `rtw_` drivers share one common part, which is what reports Wi-Fi AP support, so the capability applies to all of them. It was confirmed on hardware for the RTL8821CU and the RTL8812AU. The same common part handles broadcast and multicast traffic in a way that needs two settings of the access point daemon, see Section 5.3.

Warning: The last two rows are the trap. Adapters based on the MT7601U or driven by rtl8xxxu report Wi-Fi Station mode only. They are widely sold, inexpensive and work well as Wi-Fi Stations, but they cannot be used for the setup described in this document. For rt2800usb the capability depends on the individual chip, which has to have a beacon queue.

2.3 Checking an adapter

Once the wireless interface has been handed to the container (Section 4), ask the driver directly instead of relying on the table. The relevant line is `Supported interface modes`, and it has to contain `AP`:

```
# iw phy phy0 info
Wiphy phy0
        Supported interface modes:
                 * IBSS
                 * managed
                 * AP
                 * AP/VLAN
                 * monitor
                 * P2P-client
                 * P2P-GO
        valid interface combinations:
                 * #{ managed } <= 1,
```

```
                    #{ AP, P2P-client, P2P-GO } <= 1,
                    total <= 2, #channels <= 1
```

The `valid interface combinations` line is worth reading as well. The example above permits exactly one Wi-Fi AP, and it permits a Wi-Fi Station connection alongside it only on the same channel. An adapter cannot serve a Wi-Fi AP on one channel and be a Wi-Fi Station on another at the same time.

2.4 Frequency bands

Both bands are usable. 2.4 GHz offers the best compatibility with Wi-Fi Stations and range, 5 GHz offers the higher data rate and a less crowded spectrum. Which channels are available is decided by the regulatory domain, see Section 5. Channels marked for radar detection (DFS) require functionality that the Wi-Fi AP daemon does not provide on most drivers, so plan for a non-DFS channel.

Choose the channel after looking at the networks around. Before the Wi-Fi AP daemon runs, the interface can scan:

```
ip link set dev wlx58044f6d24b0 up
iw dev wlx58044f6d24b0 scan | grep -E 'freq:|signal:|SSID:'
```

Prefer the channel that the fewest and the weakest networks use. On 2.4 GHz this is one of the channels 1, 6 and 11, which do not overlap.

3 Preparing the Debian Container

The Debian container provides the userland: the wireless configuration tool and the Wi-Fi AP daemon. Install and start the container as described in the *Debian Container User Guide* [1], then open an SSH session to it. Container names follow the pattern `<container-name>-<periMICA-name>.local`:

```
ssh root@debian-perimica-sernm.local
```

For repeated access, deploy an SSH key instead of using a one-time password.

3.1 Installing the packages

The container resolves names through the periMICA and reaches the Debian archive, so the packages are installed with the ordinary package manager. Two packages are needed, and the netlink libraries they depend on are pulled in automatically:

hostapd	Wi-Fi AP daemon
iw	wireless configuration tool

Table 3: Packages required in the container

```
apt-get update
apt-get install -y hostapd iw
```

Note: On a periMICA without access to the Debian archive, download the two packages and their dependencies on a workstation, transfer them with `scp`, and install them with `dpkg -i`, libraries first. The dependencies are the netlink libraries `libnl-3-200`, `libnl-genl-3-200` and `libnl-route-3-200`.

Warning: The Debian package installs a systemd service for the Wi-Fi AP daemon but leaves it masked, because service startup is suppressed while a package is unpacked into a container. This applies to both installation methods. Autostart therefore has to be enabled explicitly, see Section 5.

3.2 Keeping the container self-contained

Everything the Wi-Fi AP needs beyond the driver is placed inside the container: the packages above and the configuration file of the Wi-Fi AP daemon. Driver and adapter firmware are part of the periMICA firmware (Section 2). Nothing is written to the base system. A container

exported in this state can be installed on another periMICA and brings its wireless configuration with it. The wireless interface has to be handed to it on that periMICA (Section 4), and because the interface name contains the MAC address of the adapter, the name in the configuration has to be adapted when the adapter changes.

4 Handing the Radio to the Container

The driver runs in the kernel, so the radio device it creates initially belongs to the base system, not to the container. It has to be moved into the container's network namespace. After the move the wireless interface exists only inside the container, and the base system has no wireless interface at all. Stopping the container returns the radio to the base system.

4.1 Moving the radio

Right-click on the container and select *Settings* from the *Options* dropdown menu. In the Settings dialog, select the *Details* dropdown. Select the wireless interface from the *Additional Interface* dropdown list. Wireless interfaces start with *wlx*, suffixed with their MAC address. Click the *Apply* button to move the interface into the container. The interface is now available inside the container, and the base system has no wireless interface at all.

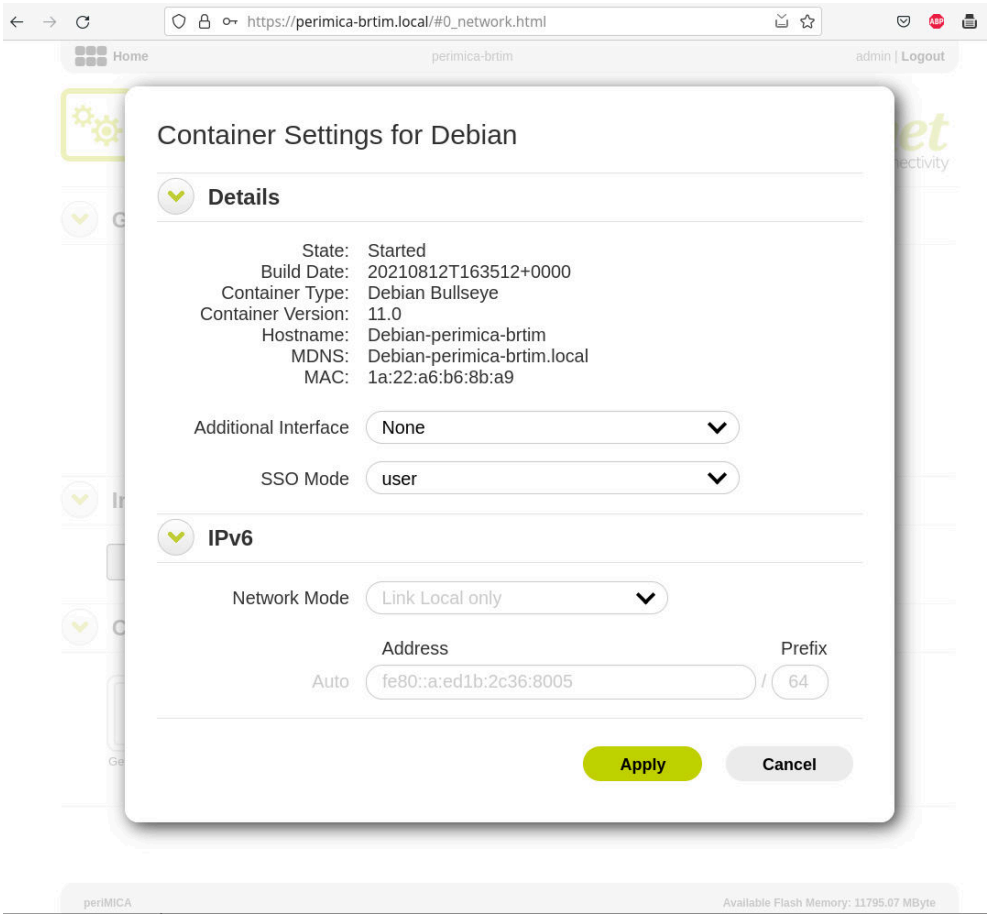


Figure 1: periMICA Container Settings

5 Running the Wi-Fi AP

With the radio inside the container, the Wi-Fi AP is a matter of the regulatory domain and one configuration file.

5.1 Regulatory domain

A Wi-Fi AP may only transmit where transmission is permitted, and the kernel enforces this. Without a regulatory domain the radio operates in world roaming mode, in which all 5 GHz channels and the upper 2.4 GHz channels are marked as “no initiating radiation”. A Wi-Fi AP can then use channels 1 to 11 only, and 5 GHz not at all. This is not a warning that can be ignored, the daemon simply refuses those channels.

The kernel loads the regulatory database itself. The database is signed, and the periMICA kernel verifies the signature against certificates compiled into it. Set the domain and verify that the database is in effect:

```
# iw reg set DE
# iw reg get
global
country DE: DFS-ETSI
        (2400 - 2483 @ 40), (N/A, 20), (N/A)
        (5150 - 5250 @ 80), (N/A, 23), (N/A), NO-OUTDOOR, AUTO-BW
        (5250 - 5350 @ 80), (N/A, 20), (0 ms), NO-OUTDOOR, DFS, AUTO-BW
        (5470 - 5725 @ 160), (N/A, 26), (0 ms), DFS
        (5725 - 5875 @ 80), (N/A, 13), (N/A)
```

A domain other than `country 00` in the output is the proof that the database was loaded. If it stays at `country 00`, the database was not loaded.

Note: The regulatory domain belongs to the kernel, and there is one for the whole periMICA. Setting it in one container sets it for the base system and for every other container as well.

5.2 Configuring the Wi-Fi AP

A minimal configuration for the 2.4 GHz band with WPA2 protection. Wireless interfaces are named `wlx` followed by the MAC address of the adapter (Section 4). The examples use `wlx58044f6d24b0`, replace it with the name of your interface:

```
interface=wlx58044f6d24b0
driver=nl80211
ssid=periMICA-AP
country_code=DE
ieee80211d=1
hw_mode=g
channel=6
ieee80211n=1
```

```
wmm_enabled=1
auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_passphrase=<passphrase>
ctrl_interface=/var/run/hostapd
```

To admit WPA3 Wi-Fi Stations beside WPA2 ones, replace the key management line and add management frame protection, which WPA3 requires and WPA2 Wi-Fi Stations may use:

```
wpa_key_mgmt=WPA-PSK SAE
ieee80211w=1
sae_require_mfp=1
```

For the 5 GHz band on a channel without radar detection, replace the band and channel settings and add the parameters for 80 MHz channel width. The centre frequency index belongs to the channel group, index 42 covers channels 36 to 48:

```
hw_mode=a
channel=36
ieee80211ac=1
ht_capab=[HT40+]
vht_capab=[SHORT-GI-80]
vht_oper_chwidth=1
vht_oper_centr_freq_seg0_idx=42
```

Warning: The capability parameters have to match what the adapter reports. A capability listed in the configuration but not supported by the adapter makes the daemon refuse to start. The supported capabilities are part of the `iw phy` output shown in Section 2.

5.3 Broadcast and multicast with rtw drivers

With an adapter on one of the `rtw_` drivers (Table 2), a Wi-Fi AP that is bridged with the periMICA's network can lose its Wi-Fi Stations after a minute or two of traffic. The kernel then reports, many times in a row:

```
rtw_8821cu 1-1.1:1.2: error beacon valid
rtw_8821cu 1-1.1:1.2: failed to download drv rsvd page
```

The Wi-Fi AP stops updating its beacon. Wi-Fi Stations lose the network, and afterwards they do not get past authentication. The cause is in the driver: it sends every broadcast and multicast frame, such as the mDNS and SSDP traffic of the network, through a queue that the adapter empties only at a DTIM beacon. On a busy network this queue uses up the transmit memory of the adapter, and the beacon can no longer be updated. The firmware of the adapter

is not the cause. A correction of the driver is in preparation in the wireless development tree of the Linux kernel.

Two settings avoid the problem. The daemon sends multicast to each Wi-Fi Station as unicast, in its own queue, and announces a DTIM with every beacon. Add them to the configuration:

```
dtim_period=1
multicast_to_unicast=1
```

Keep the default beacon interval and do not set `beacon_int`. A longer interval empties the queue even less often.

In a test with an RTL8812AU adapter under sustained traffic, the Wi-Fi AP without these settings lost its Wi-Fi Station after about one minute. With them it ran for 15 minutes without a single driver message.

Note: With `multicast_to_unicast` every multicast frame is sent once per Wi-Fi Station. For the few Wi-Fi Stations of a periMICA Wi-Fi AP this costs little airtime, and each Wi-Fi Station receives the frames at its own data rate.

5.4 Starting and verifying

```
hostapd -B -t -f /var/log/hostapd.log /root/wifi-ap/hostapd.conf
```

The log has to reach the state `AP-ENABLED`. The interface then reports type `AP` together with the network name and the channel in use:

```
# iw dev wlx58044f6d24b0 info
Interface wlx58044f6d24b0
        ssid periMICA-AP
        type AP
        channel 6 (2437 MHz), width: 20 MHz, center1: 2437 MHz
        txpower 20.00 dBm
```

Associated Wi-Fi Stations are listed with `iw dev wlx58044f6d24b0 station dump`, and the association and authentication of each Wi-Fi Station is recorded in the log file.

5.5 Starting automatically

The systemd service of the Wi-Fi AP daemon is masked after installation (Section 3). It reads its configuration from **/etc/hostapd/hostapd.conf**, and it does not start while that file is missing or empty. Copy the configuration there, then unmask and enable the service:

```
cp /root/wifi-ap/hostapd.conf /etc/hostapd/hostapd.conf
systemctl unmask hostapd
systemctl enable --now hostapd
```

Note: Setting `DAEMON_CONF` in **/etc/default/hostapd** to another file does not help: the Debian package marks it as deprecated, and the service still waits for **/etc/hostapd/hostapd.conf**.

5.6 Bridged

An associated Wi-Fi Station still needs an address and a path to whatever it wants to reach. The Wi-Fi AP daemon does not provide either, it only handles the radio link. Two arrangements are common: bridged, described here, and routed (Section 5.7). The choice depends on whether the Wi-Fi Stations are meant to be part of the wired network or to stay separate.

The wireless interface and the container's wired interface are joined by a bridge inside the container. The Wi-Fi Stations then appear on the periMICA network as if they were cabled, and they are configured by whatever already configures that network: the router advertisements of the site router, or its DHCP server. No additional service is required in the container, and the periMICA and its containers remain reachable for the Wi-Fi Stations under their usual names.

The bridge is set up before the Wi-Fi AP daemon starts, and it takes over the addresses of the wired interface. The link-local address is among them, and it has to stay the same, because the periMICA knows the container by it:

```
ip addr show dev eth0
ip link add name br0 type bridge
ip link set dev br0 address $(cat /sys/class/net/eth0/address)
echo 1 > /proc/sys/net/ipv6/conf/br0/addr_gen_mode
echo 0 > /sys/class/net/br0/bridge/multicast_snooping
ip addr flush dev eth0
echo 1 > /proc/sys/net/ipv6/conf/eth0/disable_ipv6
ip link set dev eth0 master br0
ip link set dev eth0 up
ip addr add <address of eth0> dev br0
ip link set dev br0 up
```

Add every address the first command listed for `eth0`, except those marked `dynamic`, which the bridge obtains again from the router advertisements, and add the default routes of `eth0` to `br0`. The wired interface carries no address of its own any more, so IPv6 is switched off on it. Three of the settings are not optional:

- **The MAC address of the wired interface.** A bridge takes the lowest MAC address of its ports. Without a fixed address it changes when the wireless interface joins, and with it the addresses the container derives from it.
- **No link-local address of its own** (`addr_gen_mode`). The address taken over from `eth0` stays the only one, and it is the one the container is announced with.

- **Multicast snooping off.** mDNS and the router advertisements then reach every Wi-Fi Station, also when nobody on the network sends the queries that snooping relies on.

The Wi-Fi AP daemon adds the wireless interface to the bridge itself. Add one line to its configuration (Section 5):

```
bridge=br0
```

The commands above do not persist. Run them at every start of the container before the Wi-Fi AP daemon, for example from a systemd unit of type `oneshot` that is ordered before `hostapd.service`.

Warning: An interface loses its own address when it is added to a bridge. Do not reconfigure the wired interface over an SSH session that runs through that same interface. Use the peri-MICA console, or arrange for the previous configuration to be restored automatically if the session is lost.

5.7 Routed

The wireless interface keeps its own address range, and the container routes between the Wi-Fi Stations and the wired network. This isolates the Wi-Fi Stations, which is the more appropriate arrangement when the Wi-Fi AP serves service technicians or visitors rather than plant equipment.

This variant needs one additional service in the container to configure the Wi-Fi Stations, a router advertisement daemon for IPv6 or a DHCP server for IPv4, and a decision about the address range to be used. If the Wi-Fi Stations are to reach the wired network, forwarding has to be enabled and address translation configured.

5.8 Without address configuration

For a first test neither is necessary. Assign a static address to the wireless interface, and configure the Wi-Fi Station statically as well. This proves the radio link, the authentication and the data path without introducing a second moving part.

6 Running as a Wi-Fi Station

The same adapter can also connect the periMICA to an existing Wi-Fi® network as a Wi-Fi Station, for example to give a machine without a cable to the plant network an uplink. Adapter, container and radio handover are the same as for the Wi-Fi AP, see Section 2 to Section 4. Only the daemon changes: the Wi-Fi AP daemon is replaced by a supplicant, which authenticates the container at the Wi-Fi AP of the existing network.

This chapter assumes that the Wi-Fi AP the container connects to configures its Wi-Fi Stations: IPv4 through DHCP, and IPv6 through router advertisements. The container then only has to accept that configuration, no address planning is needed for the wireless side.

As in Section 5, the examples use `wlx58044f6d24b0` as the name of the wireless interface. Replace it with the name of your interface.

6.1 Adapter requirements

Wi-Fi Station operation corresponds to the mode `managed` in the `Supported interface modes` of the adapter (Section 2). All drivers listed in Table 2 support it, including rtl8xxxu and mt7601u, which cannot be used as a Wi-Fi AP.

Note: An adapter that serves a Wi-Fi AP and is a Wi-Fi Station at the same time has to do both on the same channel, see Section 2. The channel is then dictated by the network the Wi-Fi Station connects to.

6.2 Installing the packages

wpasupplicant	supplicant, authenticates the Wi-Fi Station at the Wi-Fi AP
isc-dhcp-client	DHCP client for IPv4
iw	wireless configuration tool

Table 4: Packages required in the container for Wi-Fi Station operation

```
apt-get update
apt-get install -y wpasupplicant isc-dhcp-client iw
```

IPv6 needs no additional package. The kernel evaluates the router advertisements and configures address and default route by itself.

6.3 Configuring the connection

Set the regulatory domain as described in Section 5. Without it, the Wi-Fi Station does not use the channels that are reserved for a specific domain, and networks on those channels remain invisible.

A minimal configuration for a network with WPA2 protection, stored as **/root/wifi-sta/wpa_supplicant.conf**:

```
ctrl_interface=/var/run/wpa_supplicant
country=DE

network={
    ssid="<network name>"
    key_mgmt=WPA-PSK
    psk=<hashed passphrase>
}
```

The hashed passphrase is generated with `wpa_passphrase`, so that the passphrase itself is not stored in the container:

```
wpa_passphrase "<network name>" "<passphrase>"
```

Note: For a network with WPA3 protection, set `key_mgmt=SAE` and `ieee80211w=2` in the `network` block. This variant was not validated with the adapter in Table 1.

6.4 Connecting and obtaining addresses

Start the supplicant, then the DHCP client:

```
wpa_supplicant -B -i wlx58044f6d24b0 -c /root/wifi-sta/wpa_supplicant.conf \
    -f /var/log/wpa_supplicant.log
dhclient -4 wlx58044f6d24b0
```

The log has to reach the state `CTRL-EVENT-CONNECTED`. The link then reports the Wi-Fi AP and the network name:

```
# iw dev wlx58044f6d24b0 link
Connected to <bssid> (on wlx58044f6d24b0)
        SSID: <network name>
        freq: 2437
        signal: -52 dBm
```

IPv4 address and default route come from the DHCP server, the IPv6 address and default route from the router advertisements:

```
# ip addr show dev wlx58044f6d24b0
# ip route
default via <IPv4 router> dev wlx58044f6d24b0
# ip -6 route
default via fe80::<router> dev wlx58044f6d24b0 proto ra metric 1024 expires 1798sec
```

Warning: The DHCP client replaces **/etc/resolv.conf** with the name servers of the wireless network. If the container should keep resolving names through the periMICA, remove `domain-name-servers` and `domain-search` from the `request` statement in **/etc/dhcp/dhclient.conf**.

If the wired side of the container also provides a default route, the container has two. Traffic follows the route with the lowest metric, so decide which uplink is preferred and set the metrics accordingly.

6.5 Starting automatically

The supplicant package provides the systemd template service `wpa_supplicant@.service`, which reads **/etc/wpa_supplicant/wpa_supplicant-<interface>.conf**. Copy the configuration file there and enable the service for the wireless interface. As with the Wi-Fi AP daemon (Section 3), check that the service is not masked. The DHCP client has to be started after the supplicant, for example by the network configuration of the container.

6.6 Routing the wired network over the wireless uplink

The container can act as the router for devices on its wired side that have no uplink of their own. The Wi-Fi AP of the wireless network configures the container only. It does not know the devices behind it and has no route to them. The container therefore translates their addresses to its own on the wireless interface, so that the wireless network sees the container only.

Note: A bridge, as described in Section 5.6 for the Wi-Fi AP, is not an option here. A Wi-Fi Station interface can only send frames with its own address, so the wired devices cannot be bridged onto the wireless network.

The arrangement needs an address range for the wired side, a service that configures the wired devices, forwarding, and address translation:

- **IPv4:** a private range on the wired interface that differs from the range of the wireless network, for example `192.168.50.0/24`, a DHCP server for that range, and address translation on the wireless interface.
- **IPv6:** the router advertisements of the wireless network provide one prefix for the wireless link only, and a prefix of that size cannot be divided further for a second link. The wired side therefore gets a unique local prefix of its own (RFC 4193), announced by router advertisements, and the container translates it on the wireless interface as for IPv4.

Link-local communication on the wired side, and with it the mDNS names of the periMICA and the Perinet devices, is not affected by this.

Warning: An interface that forwards ignores router advertisements unless it is told otherwise. Enabling IPv6 forwarding in the container therefore makes the wireless interface lose its IPv6

address and default route as soon as they expire. Set `accept_ra` of the wireless interface to 2 before forwarding is enabled.

```
sysctl -w net.ipv6.conf.wlx58044f6d24b0.accept_ra=2
sysctl -w net.ipv6.conf.all.forwarding=1
sysctl -w net.ipv4.ip_forward=1
```

The wired interface receives an address from each range. The unique local prefix below is an example, generate a random one for each installation:

```
ip addr add 192.168.50.1/24 dev eth0
ip addr add fd12:3456:789a:1::1/64 dev eth0
```

One service, `dnsmasq`, provides both the DHCP server for IPv4 and the router advertisements for IPv6 on the wired interface, and it forwards name queries of the wired devices to the name servers of the container:

```
interface=eth0
bind-interfaces
dhcp-range=192.168.50.100,192.168.50.199,12h
enable-ra
dhcp-range=::,constructor:eth0,ra-only
```

Address translation and forwarding rules are set with `nftables`. Only connections that start on the wired side are forwarded, so the wireless network cannot open connections to the wired devices:

```
table ip nat {
    chain postrouting {
        type nat hook postrouting priority 100; policy accept;
        oifname "wlx58044f6d24b0" masquerade
    }
}
table ip6 nat {
    chain postrouting {
        type nat hook postrouting priority 100; policy accept;
        oifname "wlx58044f6d24b0" masquerade
    }
}
table inet filter {
    chain forward {
        type filter hook forward priority 0; policy drop;
        ct state established,related accept
        iifname "eth0" oifname "wlx58044f6d24b0" accept
    }
}
```

Warning: Run the DHCP server and the router advertisements only on a wired segment on which the container is the only router. On a segment that already has a site router or a DHCP server, a second one disturbs the configuration of every device on it, including the periMICA itself.

7 Troubleshooting

Symptom	Cause and remedy
Interface not offered in the web interface	The base system did not accept the adapter. Check that its driver is part of the periMICA firmware and that the periMICA runs firmware 24-05 or later, see Section 2.
The daemon refuses to start	Either the adapter does not support Wi-Fi AP operation (Section 2), or the configuration declares a capability the adapter does not have.
Only channels 1 to 11, no 5 GHz	No regulatory domain in effect. Check that `iw reg get` reports a country other than `country 00`.
`Cannot open RFKILL control device`	Expected in a container and not an error. The daemon continues and reaches `AP-ENABLED`. The radio is not blocked, which can be confirmed in **/sys/class/rfkill**.
As a Wi-Fi Station, the periMICA does not connect, log shows `WRONG_KEY`	The passphrase does not match the network. Regenerate the hashed passphrase, see Section 6.
As a Wi-Fi Station, the periMICA loses its IPv6 address after forwarding was enabled	Router advertisements are ignored on a forwarding interface. Set `accept_ra` of the wireless interface to 2.
Wi-Fi Stations drop after a minute or two, the kernel reports `error beacon valid`	Broadcast and multicast traffic of the network uses up the transmit memory of an adapter on an `rtw_` driver. Add the settings of Section 5.3.
The daemon does not start again, `Device or resource busy`, the kernel reports `failed to download firmware`	The adapter got stuck when the daemon stopped (observed with the RTL8821CU). Unplug and replug the adapter, or take it off the USB bus and back from the container, as shown below the table.
Wireless interface missing after the container restarted itself	The periMICA hands the interface over when it starts the container. Restart the container from the periMICA web interface, or unplug and replug the adapter.
Radio is not `phy0`	The kernel assigns the number at runtime. Read it with `iw dev` instead of assuming it.

Table 5: Common problems

A stuck adapter does not need to be unplugged. Taking its USB device off the bus and back, as unplugging and replugging would, makes the kernel enumerate it anew and load its firmware

again. The periMICA then hands the new interface to the container. Replace the interface name with the name of your interface:

```
# dev=$(readlink -f /sys/class/net/wlx58044f6d24b0/device/..)
# echo 0 > $dev/authorized; sleep 2; echo 1 > $dev/authorized
```

8 Contact & Support

For customer support, please call us at **+49 30 863 206 701** or send an e-mail to *support@perinet.io*.

For complete contact information visit us at www.perinet.io

A List of Figures

1 periMICA Container Settings 12

B List of Tables

1 The adapter used for this Application Note 7
2 USB wireless drivers and their Wi-Fi AP support 8
3 Packages required in the container 10
4 Packages required in the container for Wi-Fi Station operation 18
5 Common problems 23

C Glossary

Wi-Fi AP Wi-Fi access point: operates a Wi-Fi network that Wi-Fi Stations associate with (interface mode AP). 5–10, 13–20, 23

Wi-Fi Station Wi-Fi station: associates with a Wi-Fi AP and uses its network (interface mode managed). 5, 7–9, 14–18, 20, 23

D References

[1] Perinet GmbH. periMICA Debian Container User Guide. PRN.100.462. https://docs.perinet.io/PRN100462%20-%20Debian%20Container%20User%20Guide.pdf.

[2] Perinet GmbH. periMICA Wi-Fi Bridge Container. https://gitlab.com/perinet/periMICA-container/WiFiBridge.

[3] Dr.-Ing. Karsten Walther. “Communication Method for IoT Nodes or IoT Devices in a Local Network”. Pat. 102021111841. Perinet GmbH. https://patents.google.com/patent/EP4335078A1/en. Mar. 13, 2024. URL: http://v3.espacenet.com/textdoc?IDX=EP4335078.

[4] Dr.-Ing. Karsten Walther. “Internet-of-Things Module”. Pat. 102019126341. Perinet GmbH. https://patents.google.com/patent/EP4038350A1/en. Aug. 10, 2022. URL: http://v3.espacenet.com/textdoc?IDX=EP4038350.

[5] Dr.-Ing. Karsten Walther. “Method for Identifying Network Services in a Network Having Internet-of-Things Network Subscribers”. Pat. 102019126486.3. Perinet GmbH. https://patents.google.com/patent/EP4038863A1/en. Aug. 10, 2022. URL: http://v3.espacenet.com/textdoc?IDX=EP4038863.

E List of Patents

[3] Dr.-Ing. Karsten Walther. “Communication Method for IoT Nodes or IoT Devices in a Local Network”. Pat. 102021111841. Perinet GmbH. https://patents.google.com/patent/EP4335078A1/en. Mar. 13, 2024. URL: http://v3.espacenet.com/textdoc?IDX=EP4335078.

[4] Dr.-Ing. Karsten Walther. “Internet-of-Things Module”. Pat. 102019126341. Perinet GmbH. https://patents.google.com/patent/EP4038350A1/en. Aug. 10, 2022. URL: http://v3.espacenet.com/textdoc?IDX=EP4038350.

[5] Dr.-Ing. Karsten Walther. “Method for Identifying Network Services in a Network Having Internet-of-Things Network Subscribers”. Pat. 102019126486.3. Perinet GmbH. https://patents.google.com/patent/EP4038863A1/en. Aug. 10, 2022. URL: http://v3.espacenet.com/textdoc?IDX=EP4038863.

F Revision History

Revision	Date	Author(s)	Description
1	2026-11-04	shoe, dber	Initial Release