perplexity

Yes kernel-level intervention (e.g., for research or prototyping), consider forking the kernel and making your changes directly in cfg80211/mac80211 and the relevant driver.

first let select raspberry pi 4

To enable **kernel-level intervention for custom wireless/P2P interface management** on a Raspberry Pi 4, you must build and modify the kernel source directly. This approach lets you bypass the cfg80211 export restrictions and implement new logic in the wireless stack or driver. Here's how you can get started:

1. Obtain and Prepare the Raspberry Pi 4 Kernel Source

- **Clone the official Raspberry Pi kernel source:**

  ```
  sudo apt install git
  git clone --depth=1 https://github.com/raspberrypi/linux
  cd linux
  ```

  This gives you the latest Raspberry Pi kernel source[1] [2].

- **Install build dependencies:**

  ```
  sudo apt install bc bison flex libssl-dev make
  ```

  For cross-compilation, also install:

  ```
  sudo apt install crossbuild-essential-arm64
  ```

- **Load the default Pi 4 configuration:**

  ```
  make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- bcm2711_defconfig
  ```

  You can further customize with `make nconfig` or `make menuconfig` if needed[2].

2. Modify the Wireless Stack or Driver

- **Edit cfg80211/mac80211 or your driver source code** within the kernel tree.
  - You can add new exported functions, hooks, or even custom Netlink commands for your P2P interface management.

- For example, you could modify `net/wireless/` (cfg80211) or `drivers/net/wireless/` (for your specific chip, e.g., brcmfmac for Pi 4) to add logic for creating/managing parallel P2P interfaces.

- **If you want to export internal symbols** (like `cfg80211_iterate_wiphy`), add them to the relevant source files with `EXPORT_SYMBOL()` or `EXPORT_SYMBOL_GPL()`.
  Note: This is only safe in your custom kernel and not recommended for upstream or production use.

3. Build the Custom Kernel

- **Build the kernel and modules:**

```
make -j6 Image.gz modules dtbs
```

  (For 64-bit; for 32-bit use `zImage`)[1] [2].

- **Install the new kernel and modules on your Raspberry Pi 4:**
  - Copy the kernel image and device tree files to `/boot/`.
  - Install modules to `/lib/modules/` as needed[3].

- **Reboot to use your new kernel:**

```
sudo shutdown -r now
```

4. Test and Iterate

- After rebooting, your kernel will include your custom logic for P2P interface management.
- Use kernel logs (`dmesg`) and userspace tools to verify your changes.

Summary Table: Key Steps

Step	Command/Action	Reference
Clone kernel source	`git clone --depth=1 https://github.com/raspberrypi/linux`	[1] [2]
Install deps	`sudo apt install bc bison flex libssl-dev make ...`	[1] [2]
Load config	`make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- bcm2711_defconfig`	[2]
Modify code	Edit cfg80211/mac80211 or driver source	
Build kernel	`make -j6 Image.gz modules dtbs`	[1] [2]
Install & reboot	Copy files to `/boot/`, reboot Pi	[3]

Key Points

- **This approach gives you full control** over the wireless stack, allowing you to implement and export any functionality you need for research or prototyping.
- **All changes are local to your custom kernel** and will not be portable to stock kernels without similar modifications.
- **Be prepared for debugging and maintenance**, as kernel modifications can introduce instability.

For detailed, official instructions on building and customizing the Raspberry Pi kernel, see the [Raspberry Pi documentation][1] [2] [3].

If you need guidance on specific code changes for P2P interface management, let me know your target driver (e.g., brcmfmac) and the behavior you want to implement.

⁂

1. https://www.raspberrypi.com/documentation/computers/linux_kernel.html
2. https://gist.github.com/G-UK/ee7edc4844f14fec12450b2211fc886e
3. https://sysprogs.com/VisualKernel/tutorials/raspberry/buildkernel/