OpenWRT 21.02.x with MESHdesk packages

Follow these instructions to build the MESHdesk firmware on devices capable of running OpenWRT version 21.02.x

Minimum Hardware Requirements

- The minimum hardware requirements are:
  - **8M Flash**
  - **64M RAM**
- Although the system can potentially support hardware with less resources, supporting them in 2021 is not practical any more.
- The hardware does not need a radio on it.
- Hardware without radios can be managed using **APdesk**.
- Mediatek and Atheros / Qualcomm SOC devices are supported.
- Other target systems are also supported but have not been thoroughly tested to date.
- If you are not sure if your hardware will work please visit the OpenWRT website and check. They have an ever growing list of supported hardware.
- Next we will look at the steps you need to take to get MESHdesk working with it.

Steps In Adding New Hardware

The Very First Time (Draft)

These steps you have to do **ONCE ONLY**

1. Build OpenWRT with MESHdesk firmware (MESHdesk disabled).
2. Flash your device.
3. Prepare the **wan_network** file for specific device.
4. Prepare **meshdesk** config file for specific device.
5. Prepare **captive_config.json** file for specific device.

Afterwards (Final)

1. Build OpenWRT with MESHdesk firmware containing the device specific files for
   1. **wan_network**
   2. **captive_config.json**
   3. **meshdesk**
2. Flash your device with the final built of firmware.

- So without further ado, lets get going with the first draft built.
- In this page we will take a **Xiaomi 4A 100M** Access Point as a sample unit.
- You can use the hardware of your choice and simply apply the same principles.

- Make sure you followed these instructions to prepare the environment.

Fetching the MESHdesk package

- Check out the OpenWRT-MESHdesk package from the SourceForge repository.

```
#Do this in the working directory e.g. cd 21.02.0
git clone https://github.com/RADIUSdesk/openwrt-meshdesk.git openwrt-meshdesk
```

- The package has three main components. Each one is located in a unique folder.

1. **MESHdesk** - This is the MESHdesk package which will be build by the SDK.
2. **files** - This is the override structure containing files to override during the build process.
3. **luci-app-meshdesk** - This is the Luci application used to enable central control.

Copying the three components

- The **MESHdesk** folder needs to be copied under the **package** folder (openwrt/package).

```
#cd to the working directory
cp -R ./openwrt-meshdesk/MESHdesk ./openwrt/package
```

- The **files** folder needs to sit directly under the **openwrt** folder (root level).

```
#cd to the working directory
cp -R ./openwrt-meshdesk/files ./openwrt
```

- The **luci-app-meshdesk** folder needs to be copied under the **feeds/luci/applications** folder.

```
#cd to the working directory
cp -R ./openwrt-meshdesk/luci-app-meshdesk ./openwrt/feeds/luci/applications
```

Updating the available packages

- Since we added a Luci application, we need to tell the SDK about it.
- After you copied the packages across issue the following command:

```
#cd to the working directory
cd ./openwrt
scripts/feeds update -i
#Install the package to make it visible
scripts/feeds install luci-app-meshdesk
```

- The result is that the MESHdesk Luci application will be listed as one of the available Luci applications.

- These instructions are for the **21.02.x** branch.
- If you wish to build the firmware for older versions there are some tweaks that has to be done for it to work as intended.
- These tweaks are discussed in their own dedicated Wiki page.

Select Packages To Include With Firmware

- Select the following packages when building the firmware.
- When selecting a package there are the options to build it as a module (M) or fully include it (*).
- **Make sure you select with the (*) option to fully include the package.**
- Package names in bold are required.
- The Mosquitto packages are for MQTT support.
- The Batman packages are for mesh support.

Package	Location	Comment
MESHdesk	Base system	
kmod-batman-adv	Kernel Modules → Network Support	Keep the default options
lua-mosquitto	Languages → Lua	
libiwinfo-lua	Languages → Lua	
luasocket	Languages → Lua	
libuci-lua	Libraries	
luci	Luci → Collections	
luci-compat	Luci → Modules	Needs this modules for our package **VERY IMPORTANT**
luci-app-meshdesk	Luci → Applications	Luci App to enable and disable central management
luci-theme-material	Luci → Themes	Modern theme that is easy to customize
luci-lib-httpclient	Luci → Libraries	
luci-lib-httpprotoutils	Luci → Libraries	
luci-lib-json	Luci → Libraries	
luci-lib-jsonc	Luci → Libraries	
coova-chilli	Network → Captive Portals	Select **OpenSSL** as **SSL Library**. Also select **Enable the JSON interface..** and **..Coova miniportal...**
curl	Network → File Transfer	
relayd	Network → Routing and Redirection	
wpad IEEE 802.1x Auth/Supplicant (built-in full)	Network → WirelessAPD	Un-select wpad-basic
batctl-full	Network	Un-select batctl-default
mosquitto-client-ssl	Network	Note the **CLIENT** package

- After you selected these packages you can save the configuration and issue **make** to build the firmware.

- The firmware you just built will be standard OpenWRT in effect and you can flash your hardware as with normal OpenWRT then access it on **192.168.1.1**.
    - Username and Password is **root** and **admin** for Luci and ssh.
- The next section will cover the files you have to attend to for the specific hardware tweaks.

Initial File Preparation

- Use ssh to gain access to the device in order to tweak these files.

wan_network

- Refer to the default **/etc/config/network** file.

network

```
config interface 'loopback'
    option device 'lo'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'

config globals 'globals'
    option ula_prefix 'auto'

config device
    option name 'br-lan'
    option type 'bridge'
    list ports 'eth0.1'

config interface 'lan'
    option device 'br-lan'
    option proto 'static'
    option ipaddr '192.168.1.1'
    option netmask '255.255.255.0'
    option ip6assign '60'

config device
    option name 'eth0.2'
    option macaddr '9c:9d:7e:f6:22:1c'

config interface 'wan'
    option device 'eth0.2'
    option proto 'dhcp'

config interface 'wan6'
    option device 'eth0.2'
    option proto 'dhcpv6'

config switch
```

```
    option name 'switch0'
    option reset '1'
    option enable_vlan '1'

config switch_vlan
    option device 'switch0'
    option vlan '1'
    option ports '4 2 6t'

config switch_vlan
    option device 'switch0'
    option vlan '2'
    option ports '0 6t'
```

- Next look at the **/etc/MESHdesk/configs/wan_network** file that is derived from it.

wan_network

```
config interface 'loopback'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'
    option ifname 'lo'

config interface 'lan'
    option ifname 'eth0.1'
    option type 'bridge'
    option proto 'dhcp'

config interface 'client_0'
    option proto 'dhcp'

config interface 'client_1'
    option proto 'dhcp'

config switch
    option name 'switch0'
    option reset '1'
    option enable_vlan '1'

config switch_vlan
    option device 'switch0'
    option vlan '1'
    option ports '0 6t'

config switch_vlan
    option device 'switch0'
    option vlan '2'
    option ports '4 2 6t'
```

- We took the **switch** config sections from */etc/config/network* and simply **swapped** vlan nr 1 and vlan nr 2 around.
- This means that eth0.1 is now on the **WAN** port.
- We also kept the old config style which is still supported in **21.02**
- **client_0** and **client_1** interface sections can always be kept as is.
- **lan** interface section is in actuality the **WAN** port. (This is due to the historical nature of the project and also that we support hardware with a single Ethernet port)

- Lets look at another sample file. This time the **Xiaomi 4A Gigabit Edition**. This board does not have any **switch** sections and is much simpler.

wan_network

```
config interface 'loopback'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'
    option ifname 'lo'

config interface 'lan'
    option ifname 'wan'
    option type 'bridge'
    option proto 'dhcp'

config interface 'client_0'
    option proto 'dhcp'

config interface 'client_1'
    option proto 'dhcp'
```

- Here you can see there is no **switch** sections and we specified the ifname for the **lan** interface as **wan**. Plain and simple.
- Next we will look at the */etc/config/meshdesk* file and tweak it to work with our environment and our hardware.

- There is a growing list of existing sample **wan_network** files under the */openwrt/package/MESHdesk/files/MESHdesk/configs* folder.
- They have a convention of **network_<firmware_id>** e.g. network_xiaomi_4a_100
- Simply copy that file over *openwrt/package/MESHdesk/files/MESHdesk/configs/wan_network*
- Those files will have a matching existing hardware section in the *openwrt/package/MESHdesk/files/MESHdesk/meshdesk* file

meshdesk

Tweaks For Our Hardware

- MESHdesk use the LEDs of the device it is installed on to signal about the environment
  - It signals during startup about the method it uses to try and fetch its settings at that moment.
  - If the device is used in a mesh network it will signal how many neighboring nodes it sees after startup.
  - There is also another LED used to indicate if the device has proper contact with the controller. (The LED can be either ON or OFF in such a case)
  - Finally on mesh networks we can also specify a LED that will indicate mesh traffic flowing through a node.
  - Again lets look at the **Xiaomi 4A 100M** as a sample.

```
#change directory to where the LEDs are
cd /sys/class/leds/
ls
#These are the LEDs available
blue:power    mt76-phy0     mt76-phy1     yellow:power
#turn it off
echo "0" > yellow\:power/brightness
#turn it on
echo "1" > yellow\:power/brightness
#turn it off
echo "0" > blue\:power/brightness
#turn it on
echo "1" > blue\:power/brightness
```

- In our case we can use the **yellow LED** to light up if the comms to the controller is broken.
- We can use the **blue LED** to signal during startup and neighbor counts.
- There is no extra LED so we will not define one for the mesh traffic.
- With this info we can create a hardware section in */etc/config/meshdesk*

```
config hardware 'xiaomi_4a_100'
    option morse_led '/sys/class/leds/blue:power/brightness'
    option internet_led '/sys/class/leds/yellow:power/brightness'
    option wifi_led 'led0'
```

- This have to match the value for hardware under the **settings** section

```
config settings 'settings'
    option hardware 'xiaomi_4a_100'
    option id_if 'eth0'
    option lan_up_file '/tmp/lan_up'
```

Don't make the name of the hardware section more than 14characters. Longer names break things during deployment.

- Later we will also use the value of **xiaomi_4a_100** to define the hardware on the controller.

- The final tweak for the hardware in the config file is the interface that must be used as the **id_if**.
  - It will typically be **eth0**.
  - On devices like the MT7621 based boards it will typically be **wan**.
- Since we want the **yellow LED** to be off when the comms to the controller is fine we need to check what the current setup is

```
vi /etc/MESHdesk/reporting/report_to_server.lua
#Look for this section
    if(ok_flag)then
        internetLED('0'); -- NOTE Here we can swap thme around eg make it 0
to turn off a red LED when the internet is OK
        checkForContollerReboot('1');
    else
        internetLED('1');
        checkForContollerReboot('0');
    end
```

- The following table lists some of the important items with comments

Item	Typical value	Comment
settings → hardware	xiaomi_4a_100	Must match a hw definition in the file itself
settings → id_if	eth0	eg eth0, eth1 or wan - NOT eth0.1 (for those boards its just eth0)
settings → skip_radio_0	0	set to 1 when radio0 is a 5G radio and you don't want to use it for config SSID

Tweaks For Our Environment

- The following table lists some of the important items with comments

Item	Typical value	Comment
internet1 → disabled	1	change it to 0 in order for the device to be centrally controlled
internet1 → dns	cloud.radiusdesk.com	Supply Dummy Value If Not Using DNS System
internet1 → protocol	https	Can be http or https
internet1 → ip	176.31.15.210	Fallback when FQDN does not resolve on FQDN not used

- We are nearly done. The last stop is to edit the **captive_config.json** file to fit our specific hardware.

captive_portal.json

- Edit the file */etc/MESHdesk/configs/captive_config.json*.
- This file is a JSON structure that the device uses as a reference to configure itself with a special captive portal when it is not yet managed by the controller.
- There are only two items that might need to be tweaked
  - The radio number for the 2.4G band.
  - The **ifname** for the **lan** interface (We use the WAN port in out implementation)
- With the **Xiaomi 4A 100M Edition** radio0 is the 2.4G radio so no need to tweak that item. (If the hardware has radio1 as the 2.4G band simply look for all the references to **radio0** and

make them radio1)

- See this snippet of a device which has radio1 using the 2.4G band

```
"wireless": [
            {
                "wifi-device": "radio1",
                "options": {
                    "channel": 1,
                    "disabled": 0,
                    "hwmode": "11g",
                    "htmode": "HT20"
                }
            },
            {
                "wifi-iface": "two",
                "options": {
                    "device": "radio1",
                    "ifname": "two0",
                    "mode": "ap",
                    "network": "ex_two",
                    "encryption": "none",
                    "ssid": "_Replace_",
                    "key": "",
                    "hidden": false,
                    "isolate": false,
                    "auth_server": "",
                    "auth_secret": ""
                }
            },
            {
                "wifi-iface": "web_by_w",
                "options": {
                    "device": "radio1",
                    "mode": "sta",
                    "network": "web_by_w",
                    "encryption": "psk2",
                    "key": "radiusdesk",
                    "ssid": "meshdesk_config",
                    "disabled": "1"
                }
            }
        ],
```

- With the **Xiaomi 4A 100M Edition** the **ifname** we use is **eth0.1** for the **lan** interface definition.

```
{
    "interface": "lan",
    "options": {
        "ifname": "eth0.1",
        "type": "bridge",
```

```
        "proto": "static",
        "ipaddr": "10.50.50.50",
        "netmask": "255.255.255.0"
    }
},
```

- With the **Xiaomi 4A Gigabit Edition** it will look like this

```
{
    "interface": "lan",
    "options": {
        "ifname": "wan",
        "type": "bridge",
        "proto": "static",
        "ipaddr": "10.50.50.50",
        "netmask": "255.255.255.0"
    }
},
```

- Once the tweaks are completed we can test everything out.
- The following link shows how to point the device to the controller using the GUI
- Xiaomi Access Points Running MESHdesk Firmware
- Point the device to your controller and reboot it.
- If all goes well it will show up in **Unknown Nodes**
- If it is a new hardware type add it to the controller as described here: Hardwares (Again Once Off)
- Attach A Device To Demo1 Mesh

The Final Built

- If everything on the device work as intended you can use those tweaked files to build a final version of the firmware for the specific hardware.
- Copy the files to a temporary folder on the machine where you are building the firmware.
- Use the following as a lookup for the location inside the SDK where the tweaked files need to go.

On Device	On SDK
/etc/MESHdesk/configs/wan_network	openwrt/package/MESHdesk/files/MESHdesk/configs/
/etc/config/meshdesk	openwrt/package/MESHdesk/files/MESHdesk/
/etc/MESHdesk/configs/captive_config.json	openwrt/package/MESHdesk/files/MESHdesk/configs/
/etc/MESHdesk/reporting/report_to_server.lua	openwrt/package/MESHdesk/files/MESHdesk/reporting

- This brings us to the end of the page on how to build MESHdesk firmware for specific hardware.