How to add the SFTP protocol

Materials:

- i.MX8M Plus EVK Rev. A
- USB cable type-C
- USB cable type-B
- AC Adapter EA1045CR
- Micro SD (Optional)

Software:

- Yocto Project
- Mobaxterm Personal Edition v20.2 Build 4296

This test was done on an i.MX8M Plus EVK with Linux 5.10. Hardknott, this was also tested on an i.MX6 SabreSD.

So first of all, What is the SFTP? Secure File Transfer Protocol (SFTP) is a file protocol for transferring large files over the web. It builds on the File Transfer Protocol (FTP) and includes Secure Shell (SSH) security components.

Secure Shell is a cryptographic component of internet security. SSH and SFTP were designed by the Internet Engineering Task Force (IETF) for greater web security. SFTP transfers files security using SSH and encrypted FTP commands to avoid password sniffing and exposing sensitive information in plain text. Since the client needs to be authenticated by the server, SFTP also protects against man-in-the-middle attacks.

SFTP can be handy in all situations where sensitive data needs to be protected. For example, trade secrets may not be covered by any particular data privacy rule, but it can be devastating for them to fall into the wrong hands. So a business user might use SFTP to transmit files containing trade secrets or other similar information. A private user may want to encrypt his or her communications as well.

On the prebuild images that we provide this protocol is not available so we need to do a custom build using Yocto.

1. Setup your build following the Yocto users guide.
2. Before you bitbake edit the local.conf file:

```
nxf63675@lsv07091:~/imx-yocto-bsp/imx8mp-ddr/conf$ ls
bblayers.conf  bblayers.conf.org  local.conf  local.conf.org  local.conf.sample  templateconf.cfg
nxf63675@lsv07091:~/imx-yocto-bsp/imx8mp-ddr/conf$ nano local.conf
```

   Add the following lines to your custom build:

EXTRA_IMAGE_FEATURES ?= "debug-tweaks tools-debug eclipse-debug ssh-server-openssh"
CORE_IMAGE_EXTRA_INSTALL += "openssh-sftp openssh-sftp-server"

```
MACHINE ??= 'imx8mpevk'
DISTRO ?= 'fsl-imx-xwayland'
PACKAGE_CLASSES ?= 'package_rpm'
EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
USER_CLASSES ?= "buildstats image-mklibs image-prelink"
PATCHRESOLVE = "noop"
BB_DISKMON_DIRS ??= "\
    STOPTASKS,${TMPDIR},1G,100K \
    STOPTASKS,${DL_DIR},1G,100K \
    STOPTASKS,${SSTATE_DIR},1G,100K \
    STOPTASKS,/tmp,100M,100K \
    ABORT,${TMPDIR},100M,1K \
    ABORT,${DL_DIR},100M,1K \
    ABORT,${SSTATE_DIR},100M,1K \
    ABORT,/tmp,10M,1K"
PACKAGECONFIG_append_pn-qemu-system-native = " sdl"
CONF_VERSION = "1"

DL_DIR ?= "${BSPDIR}/downloads/"
ACCEPT_FSL_EULA = "1"

# Switch to Debian packaging and include package-management in the image
PACKAGE_CLASSES = "package_deb"
EXTRA_IMAGE_FEATURES += "package-management"
EXTRA_IMAGE_FEATURES ?= "debug-tweaks tools-debug eclipse-debug ssh-server-openssh"
CORE_IMAGE_EXTRA_INSTALL += "openssh-sftp openssh-sftp-server"
```

3. Execute “ bitbake imx-image-multimedia ” and wait.
4. Deploy your image on an SD or eMMC.
   These instructions apply to SD and MMC cards although for brevity, and usually only the SD card is listed.
   For a Linux image to be able to run, four separate pieces are needed:
   • Linux OS kernel image (zImage/Image)
   • Device tree file (*.dtb)
   • Bootloader image
   • Root file system (i.e., EXT4)

   The Yocto Project build creates an SD card image that can be flashed directly. This is the simplest way to load everything needed onto the card with one command.
   A .wic image contains all four images properly configured for an SD card. The release contains a pre-built .wic image that is built specifically for the one board configuration. It runs the Wayland graphical backend. It does not run on other boards unless U-Boot, the device tree, and rootfs are changed. When more flexibility is desired, the individual components can be loaded separately, and those instructions are included here as well. An SD card can be loaded with the

individual components one-by-one or the .wic image can be loaded and the individual
parts can be overwritten with the specific components.
The rootfs on the default .wic image is limited to a bit less than 4 GB, but re-partitioning and re-loading the rootfs can increase that to the size of the card.
The rootfs can also be changed to specify the graphical backend that is used.

Carry out the following command to copy the SD card image to the SD/MMC card. Change sdx below to match the one used by
the SD card.
$ sudo dd if=<image name>.wic of=/dev/sdx bs=1M && sync

The entire contents of the SD card are replaced. If the SD card is larger than 4 GB, the additional space is not accessible.

5. Once your system booted, on Linux console configure your connection (Ethernet or wireless), in my case I connect the board wireless as the board has an 88W8997-based Wireless Modules.

```
root@imx8mpevk:~# ifconfig mlan0 up
root@imx8mpevk:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:04:9f:06:f6:72
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 00:04:9f:06:f6:73
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:47

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1306 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1306 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:80840 (78.9 KiB)  TX bytes:80840 (78.9 KiB)

mlan0     Link encap:Ethernet  HWaddr 70:66:55:9b:36:03
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
root@imx8mpevk:~# nano /etc/wpa_supplicant.conf
root@imx8mpevk:~# wpa_supplicant -B -i mlan0 -c /etc/wpa_supplicant.conf
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
root@imx8mpevk:~# [ 2038.322940] wlan: mlan0 START SCAN
[ 2041.592211] wlan: SCAN COMPLETED: scanned AP count=2
[ 2042.607316] ASSOC_RESP: Association Failed, status code = 1, error = 0xfffc, a_id = 0xffff
[ 2042.615612] IOCTL failed: 000000004a295438 id=0x20000, sub_id=0x20001 action=1, status_code=0xfffc0001
[ 2042.727145] wlan: mlan0 START SCAN
[ 2046.416421] audit: type=1006 audit(1636612201.532:3): pid=939 uid=0 old-auid=4294967295 auid=0 tty=(none) old-ses=4294967295 ses=2 res=1
[ 2047.121813] wlan: SCAN COMPLETED: scanned AP count=2
[ 2047.147203] wlan: Connected to bssid 50:XX:XX:XX:70:e0 successfully
[ 2047.253002] mlan0:
[ 2047.253009] wlan: Send EAPOL pkt to 50:XX:XX:XX:70:e0
[ 2047.269126] mlan0:
[ 2047.269132] wlan: Send EAPOL pkt to 50:XX:XX:XX:70:e0
[ 2047.276933] IPv6: ADDRCONF(NETDEV_CHANGE): mlan0: link becomes ready
[ 2047.283874] woal_cfg80211_set_rekey_data return: gtk_rekey_offload is DISABLE

root@imx8mpevk:~# iwconfig mlan0
mlan0     IEEE 802.11-DS  ESSID:"IZZI-DA62"
          Mode:Managed  Frequency=2.462 GHz  Access Point: 50:95:51:DE:70:E0
          Bit Rate:78 Mb/s   Tx-Power=24 dBm
          Retry limit:9   RTS thr=2347 B   Fragment thr=2346 B
          Encryption key:****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****-****   Security mode:open
          Power Management:off
          Link Quality=4/5  Signal level=-52 dBm  Noise level=-98 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:4651
          Tx excessive retries:13  Invalid misc:18   Missed beacon:0
```

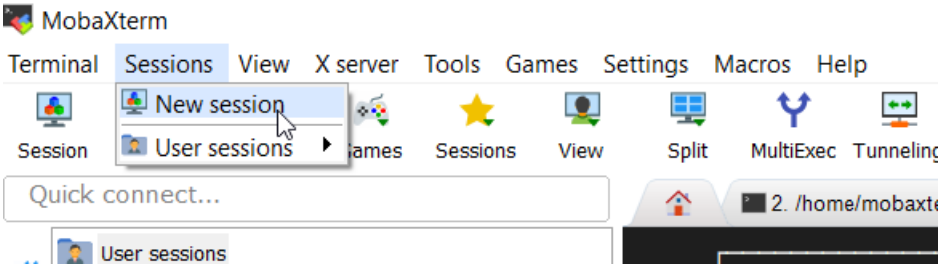
```
root@imx8mpevk:~# udhcpc -i mlan0
udhcpc: started, v1.31.0
udhcpc: sending discover
udhcpc: sending select for 192.168.0.8
udhcpc: lease of 192.168.0.8 obtained, lease time 3600
/etc/udhcpc.d/50default: Adding DNS 10.206.133.89
/etc/udhcpc.d/50default: Adding DNS 201.167.94.1
```

6. When you finish to configure your connection review the address that was assigned to you, in my case is the 192.168.0.8

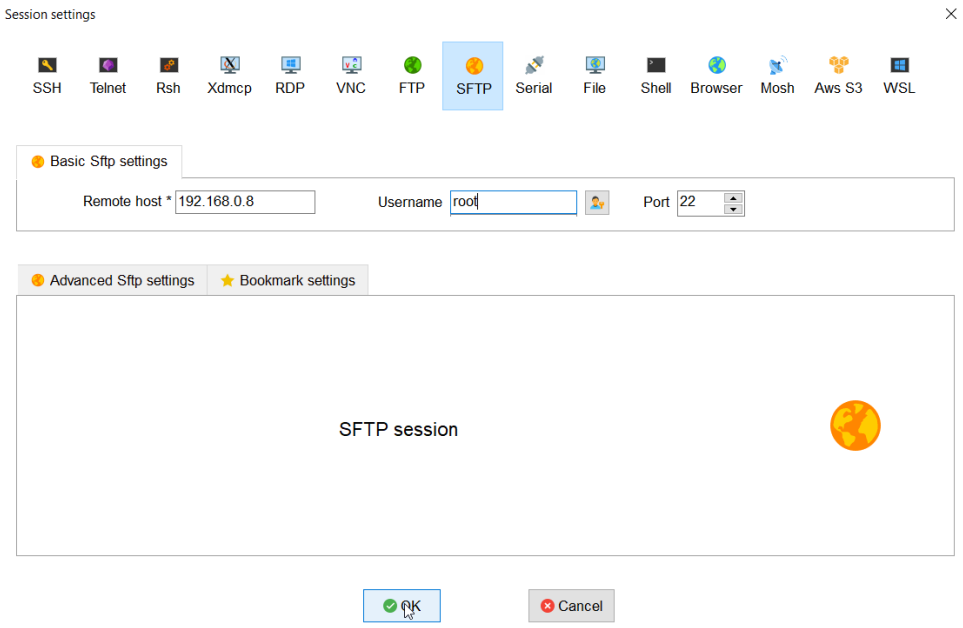
```
root@imx8mpevk:~# ifconfig mlan0
mlan0     Link encap:Ethernet  HWaddr 70:66:55:9b:36:03
          inet addr:192.168.0.8  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: 2806:310:12c:821a:7266:55ff:fe9b:3603/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1518 errors:0 dropped:0 overruns:0 frame:0
          TX packets:218 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:213019 (208.0 KiB)  TX bytes:25139 (24.5 KiB)

root@imx8mpevk:~#
```

7. Open mobaxterm and select sessions -> New sessions,



8. It will pop a new window, in that select SFTP, on remote host fill the IP address that was assigned to the board, on user name in this case is root, and the click OK:



9. Now the SFTP connection is ready! You will see something similar like below: