CONCEPTRONIC USB BLUETOOTH DONGLE CBTU
&
Linux Red Hat 8.0

HOW-TO
Installation of the Conceptronic CBTU on a Linux Red Hat 8.0 system

By Jeffrey Jongejan v1.0 01-04-2003

1. **We tested with the following:**

   - Red Hat 8.0 (psyche)
   - Linux kernel 2.4.20 (02-11-28) with the mh6 patch (03-03-22)
   - Conceptronic CBTU
   - Nokia 7650
   - Sony Ericsson P800

2. **Download the following:**

   - Latest Linux kernel from http://www.kernel.org
   - Latest Mh kernel patch from http://www.holtmann.org/linux/kernel
     Needed to patch bluetooth support into the kernel
   - bluez-libs-2.4.tar.gz, bluez-sdp-1.1.tar.gz and bluez-utils-2.3.tar.gz from http://bluez.sourceforge.net
     Bluetooth support for linux and utils to configure bluetooth
   - openobex-1.0.0.tar.gz and openobex-apps-1.0.0.tar.gz from http://openobex.sourceforge.net
     Utils needed to receive files
   - obexserver.c from http://www.frasunek.com/sources/unix/obexserver.c
     Needed to compile obexserver that is used for receiving files
   - ussp-push from http://unrooted.net/hacking/bluez-rfcomm-obex.html
     Needed to send files from the linux box to the phone

3. **Compiling the kernel with bluez support:**

   - Copy the kernel to /usr/src/ and untar the kernel with the following command:
     # tar zxfv linux-2.2.20.tar.gz
   - Copy the mh patch to /usr/src/linux-2.4.20 and apply the patch with:
     # gzip –cd patch-2.4.20-mh6.gz | patch –p1 -E
     when you get the error “file is not in gzip format” try the following command:
     # cat patch-2.4.20-mh6.gz | patch –p1 –E
   - Configure and compile your kernel and don’t forget to apply the following as modules in the bluetooth section:
     - Bluetooth subsystem support
     - L2CAP protocol support
     - SCO links support
     - RFCOMM protocol support
     - RFCOMM TTY support
     - BNEP protocol support
     - Bluetooth device drivers → HCI USB Driver
   - Edit /etc/modules.conf and add the following:
     - alias bt-proto-0 l2cap
     - alias net-pf-31 bluez
     - alias bt-proto-2 sco
     - alias bt-proto-3 rfcomm
   - Load your bluetooth driver and create /dev/rfcomm* with:
     # cd /dev && mknod rfcomm0 c 216 0 && mknod rfcomm1 c 216 1
   - A howto about kernel compiling can be found at www.linux .org in the howto section.

4. **Compile, install and configure bluez:**

   - Unpack bluez-libs-2.4.tar.gz and go to that directory. Configure, compile and install bluez-libs:
     # ./configure
     # make

make install

- Unpack bluez-utils-2.3.tar.gz and go to that directory. Configure, compile and install bluez-sdp:
  # ./configure
  # make
  # make install

- Unpack bluez-sdp-1.1.tat.gz and go to that directory. Configure, compile and install bluez-sdp:
  # ./configure
  # make
  # make install
- Now configure bluez with the file /etc/bluetooth/hcid.conf.
- If you want to change your bluetooth id you have to edit this section in hcid.conf:
  # Local device name
  # %d - device id
  # %h - host name
  name "your name (%d)"
- The default pin code is BlueZ. If you want to change this you must edit the file /etc/bluetooth/pin
- Launch hcid and sdpd
  # hcid
  # sdpd
- Check if the CBTU is up with the command:
  # hciconfig
  If the device is down you can start it this way:
  # hciconfig hci0 up

5. Compile and install openobex and obexserver

- Unpack openobex-1.0.0.tar.gz and go to that directory. Configure, compile and install:
  # ./configure
  # make
  # make install
- Unpack obexserver-apps-1.0.0 and go to that directory. Configure, compile and install as followed:
  #./configure && make
  # cd src
  # wget http://www.frasunek.com/sources/unix/obexserver.c
  # cc –o obexserver obexserver.c libmisc.a –lopenobex
  # chown root.root obexserver && cp obexserver /usr/local/bin
- Now we have to register a SDP service for openobex. Nokia uses rfcomm channel 10 and Sony Ericsson uses rfcomm channel 3. We add them as followed:
  #sdptool add –channel=3 OPUSH
  #sdptool add –channel=10 OPUSH
- If you have another brand phone you van detect the rfcomm channel as followed:
  # hcitool inq
  Here we can see the mac address of your phone.
  #sdptool browse (mac_address)
  (you can see the complete command list of sdptool with the command sdptool –help)
  Here you can find the channel number by the PUSH information
- Now start the obex server. From now you can send files from your phone to the computer.
  #obexserver

6. Compile ussp-push and upload photos to your phone.

- Unpack ussp-push.tar.gz en go to that directory. No we first have to edit obex_main.c. Open the file with an editor and change the line "custfunc.userdata = gt->userdata" to "custfunc.customdata = gt->userdata". Now save and exit
- Compile the program.
  # make
- Now we have to connect to the phone with the following command.
  # rfcomm connect 0 <Mac_adress of phone> <OBEX_PUSH Channel>
  The mac address is different for every phone and the OBEX_PUSH channel depends on the brand of the phone. Nokia uses channel 10 and Sonyericson uses channel 3 or 9.
- There now is a connection and we can send a file with the following command:

```
# ./ussp-push /dev/rfcomm0 ./<filename> <name for sending>
```
The phone should receive the file.

7. Nokia 6750 and Sony Ericsson P800.

We tested with two types of phones that supported bluetooth. The Nokia 7650 and the Sony Ericsson P800. With the Nokia we didn't experience any problems and file sending from and to the phone worked well. The Sony Ericsson on the otherhand gave some problems. File sending only worked when we send from the system to the phone. This seemd to be caused by the phone that changes the OBEX_PUSH channel once in a while.