Affix - End User Manual

Henri Ranki

Affix - End User Manual

by Henri Ranki

Table of Contents

List of Tables

Preface

Affix^{tm} is a Bluetooth protocol stack for Linux developed ad Nokia Research Center in Helsinki, Finland. It is a replacement for Bluez that is included in the kernel by default.

This manual is intended to the end users who wish to use Affix with their Bluetooth equipment. Here you find instructions for Affix installation and using Affix administration utilities. In the beginning the Bluetooth technology will be shortly introduced to help you to understand what you are doing. After the survey the Affix installation will be covered. Chapters three discuss about data security. After that chapters four and five introduces couple of Bluetooth configurations. You will be shown how to create a LAN connection between two or more computers over a Bluetooth link and in the chapter six you'll learn how to use your computer with some other Bluetooth devices like mobile phone and keyboard.

Using Affix does not assume programming or deep Linux hacking, but some experience or basic knowledge about configuring and compiling the kernel is needed. I hope you have nice time with your Bluetooth devices and Affix!

Chapter 1. Bluetooth

Idea for Bluetooth development originates to Ericsson. The main idea was to develop short distance wireless connection that would replace cable connection. The Bluetooth Special Interest Group (SIG) was founded in 1998 to implement the technology and define common de facto standard. The aim of the standard was to implement short distance wireless communication link between two or more equipments.

Technology is based on chips that have relatively low transmission power. Nevertheless the radio connection can penetrate thin walls. Connection range for class 1 chips are up to 100 m. Most of the current chips belongs to class 3 which range is up to 10 m.

When Bluetooth nodes are in range they can establish an ad hoc connection. This is called piconet. One piconet has one master node and from one to seven slaves. Individual piconets can be linked into a scatternet. In these net some nodes are member in more than one piconet.

Figure 1-1. Illustration of the scatternet

Bluetooth uses frequency 2.4GHz for communicating. The frequency spectrum is used also by other devices like wireless telephones and microwave ovens, that might otherwise disturb the transmission. To improve the quality of the connection Bluetooth uses frequency hopping technique. Bluetooth nodes are able to transmit 1 Mb/s. This is also the capacity of the piconet.

Bluetooth uses wireless LAN standard IEEE 802.11 in data transmission. This is why Bluetooth equipments can be easily integrated in TCP/IP network.

Transmission between nodes is whether synchronous or asynchronous. Synchronous is used mainly for transmitting voice and asynchronous connection is used to transmit data. Synchronous connection is symmetric and after establishing the connection both the master and the slave nodes can send packets without opening a connection again. Asynchronous connection is whether asymmetrical or symmetrical. The master node define the used bandwidth. In both cases the connection is established by the master. Connections between nodes can be whether point-to-point or point-to-multipoint connections. Special characteristics of piconets is that rather than connecting to network as in traditional LANs Bluetooth devices connect directly to another device. Connections are dynamic and can change quickly.

1.1. Protocols

Bluetooth standard uses existing protocols as much as possible. Still it defines couple of new protocols. These are L2CAP (Link Layer Control and Adaptation Protocol), SDP (Service Discovery Protocol), RFCOMM and TCS (Telephony, Control Protocol).

L2CAP operates on session control layer (in OSI model) and it makes it possible to use several different protocols on the top of itself. It is considered with segmentation and assembling the sent and received datagrams. It provides the upper layers with quality of service management. L2CAP transfers only data not audio and uses asynchronous connectionless (ACL) transfer.

SDP is used to find services near you. It is used to query services and the information how to connect with them. Services around you might change rapidly which is different from traditional networks. SDP makes this dynamic service discovery possible. It operates on the top of L2CAP protocol. SDP is also used establishing some of the (non-SDP) connections to the service.

RFCOMM provides ordinary RS-232 connection emulation. It provides both point-to-point and point-to-multipoint connections. It relies on L2CAP protocol that takes care of multiplexing the several connections over one L2CAP connection. TCS protocol defines how telephone calls are transmitted over a Bluetooth link. It is used to carry voice and data calls between Bluetooth devices.

Chapter 2. Installing Affix

Starting from version 3.0.0 Affix supports Linux kernel version 2.6 and above. These installation instructions applies to Affix 3.0.0 and later installed on Debian (sarge).

Basic Affix setup includes Bluetooth protocol stack (kernel part) and the management utilities. These two parts are installed separately. To use Affix you have to have the following packages installed:

- affix-kernel-<version>: This contains the kernel part and it should be installed first.
- openobex-1.0.1: Install this if you are going to use OBEX profile. You can download the sources from http://sourceforge.net/projects/openobex/
- affix-<version>: the sources for the end user utilities.
- devfsd: This allows you access device file system (/dev/). This is highly recommended.

Also check that the following packets are installed to your system if you use PCMCIA Bluetooth card.

- pcmcia-cs : PCMCIA device manager
- Check that your system uses yenta_socket. As root give command

  cat /etc/default/pcmcia

  This should show line "PCIC=yenta_socket". If not change the line as stated and check that your kernel has yenta support compiled as a module.

If you use Bluetooth USB stick the following package is recommended.

- hotplug: hot-plugging support for USB devices

2.1. Kernel Configuration

You have to configure you kernel before you can start installing Affix kernel module. To configure your kernel directory and use command **make menuconfig** to configure your kernel. Then,

- disable "Bluetooth support" (Bluez) support on Networking support page
- disable "USB Bluetooth TTY support" on USB support page
- enable support for USB devices
- enable support for PCMCIA device
- enable yenta_socket support
- enable support hot plug devices

- enable DEVFS device file system. This is highly recommended.
- enamble support for PPP

To configure a Personal Area Network (PAN) you have to enable the following features under Device Drivers->Networking support->Networking options->Network packet filtering->IP:Netfilter Configuration

- Iptables support
- Full NAT
- Masquerade

Read more about configuring PAN in chapter Chapter 4. After all configuration is done compile and install your kernel. Now you are ready to start installing Affix.

2.2. Installing Affix Kernel Package

Download latest affix-kernel package from http://affix.sourceforge.net/. Then log into your computer as root.

1. Unpack the package to the /usr/src/ directory.

   tar xvfz -C /usr/src affix-kernel-<version>

2. Configure Affix modules. Change to Affix kernel root directory and give command

   make config

   Answer the questions according to your hardware and system configuration. Check that the kernel directory is correct. In protocol section choose the protocols you are planning to use. If your device support Bluetooth 1.2 you should say "yes" to "BLUETOOTH 1.2 support". Do not enable debugging if you do not need it.

3. Compile Affix

   make all

4. Install Affix

   make install

* *Note! If you reinstall your kernel modules you have to also install Affix modules again (see point four above).*

2.3. Installing Affix Utilities

You should now have Affix Kernel package installed in your system. Installing utilities is very straightforward. Follow the steps below.

1. Run configuration script

   ./configure

   You can enable additional features providing '--enable-FEATURE' parameters to configure script. It accepts the following feature options:

   ```
   --enable-audio use audio (default is "yes")
   --enable-sdp use sdp (default is "yes")
   --enable-obex use obex (default is "yes")
   --enable-pan use pan (default is "yes")
   --enable-hfp use hfp (default is "no")
   --enable-debug use debug (default is "no")
   ```

2. Compile

   make

3. Install

   make install

2.4. Affix Configuration Files

To change the behavior of your Affix installation edit configuration files in /etc/affix directory. It should contain the following files: affix.conf, btsrv.conf, dhcpd.conf and pan.conf. In addition to configuration files this directory has some scripts needed to manage your Affix.

Chapter 3. Security Settings

When wireless connection is used in communication the eavesdropping is much easier. That’s why security and encryption is very important to Bluetooth devices. Bluetooth specifies security on several levels from baseband to service level. Baseband uses SAFER+ algorithms for security procedures. Bluetooth’s encryption engine requires Master nodes Bluetooth address, it’s slot clock and secret key that is shared by all the participant devices. All nodes in a piconet knows the masters device address. The secret code is called pin code.

The security level can be changed. You can choose from three level of security:

- Open - no security at all. All devices can connect.
- Link - Link level security. All the services are secured.
- Service - Securing on service level.

Security can be set to "auth" or "encrypt" depending on your need. To manage security settings use command **btctl**.

3.1. Setting Pin Codes

The process when devices create the security connection is called pairing. After pairing the data transfer between these equipments is secure. From the end user’s point of view pairing is quite simple. You just have to give the same pin code for both devices. Depending on your security settings pairing is needed before the connection can be established. Secure connection is recommended as then a third party cannot read the communication between devices. This is why Affix has pairing switched on as a default. Here is the instructions how to set pin codes.

Simplest way is to run **btsrv** on your Bluetooth computers. This way you will be prompted when the pairing is requested. When the dialog pops up on the screen just type in the pin code and press ’OK’ button.

If you are not running **btsrv** you can set pin codes manually with **btctl** command:

- Use **btclt** to find out and list the Bluetooth devices around your computer.
- To add pin code:

  ```
  btctl addpin <address_of_the_remote_peer> <pin_code>
  ```

You can create a default pin code replacing the remote peer address above with keyword *default*. Give this default pin code for the remote device when the code is prompted.

Chapter 4. Personal Area Network

This chapter explain how to set up a network connection over a Bluetooth link between two or more Bluetooth capable computers. Personal area network (PAN) is a piconet (see chapter 1) where two or more Bluetooth capable computers participate.

Host that allows other computers to connect with itself over Bluetooth connection is called Network Access Point (NAP) or Group Network (GN). You have two options to set a NAP up. You can use routing option with masquerading or you can set up a bridge. In the following sections both configurations are introduced.

GN is a configuration when there is no IP forwarding or routing features used to connect piconet participants to a LAN. In GN configuration hosts are connect together over Bluetooth link. You can think that NAP is GN expanded with a LAN connection. Thus setting up a GN is a subset of setting up a NAP which is explained here.

Note! When you compile your kernel on NAP remember to enable the following features under Device Drivers->Networking support->Networking options->Network packet filtering->IP:Netfilter Configuration

- Iptables support
- Full NAT
- Masquerade

Make sure you have IP forwarding enabled. Give the following command as root user in the console:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Note! Currently Affix does not support connection encrypting. To establish a secure connection use LAN over RFCOMM instead. See the following chapter.

4.1. Setting up a Router

To configure your computer to act as network access point (NAP) do the following steps. These steps instruct you to set your computer up using a router configuration.

1. Edit Affix configuration file (/etc/affix/affix.conf). Check that PAN role "nap" is chosen. Comment other roles out.

   Also choose "router" as pan scheme uncommenting the line pan_scheme="router" and commenting other schemes out.

2. Restart PAN giving commands:

   ```
   btpan stop
   btpan init nap
   ```

   Now you are able to connect this computer from the others. To allow TCP/IP traffic over just created PAN follow also the following steps. You need to restart PAN only this time. Next time you reboot your computer PAN is initialized automatically as NAP.

3. Set up the pan0 interface if it is not set up yet.

   ```
   ifconfig pan0 192.168.0.1 up
   ```

   Now pan0 interface should show up in the interface list you get typing command **ifconfig** without parameters.

4. You probably want to set up DHCP server in your NAP. Check that you have *dhcpd* installed in your system and then start the daemon:

   ```
   dhcpd -q -cf /etc/affix/dhcpd.conf pan0
   ```

   Now you have configured an access point for your group network (GN). If you have a network connection in your computer you probably want to share it with others and configure your computer as a NAP. If so, proceed to step five. If GN is enough you can skip the following point.

5. To use your host as a router to the network your computer is connected you should allow masquerading. To allow masquerading edit your iptables:

   ```
   iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -j MASQUERADE
   ```

   Change the IP address above to correspond the address and IP mask you have chosen to your NAP address.

Now you are ready to set up the client (PANU) computers. See instructions in the third section in this chapter.

When you boot your NAP server you do not need to repeat all the instruction above. All you need to do is allow IP forwarding as instructed in the previous section.

4.2. Setting up a Bridge

Bridge is an networking equipment that delivers IP packets from an network to another. To connect your PAN and LAN via bridge follow the instructions below.

1. Edit Affix configuration file (/etc/affix/affix.conf). Check that PAN role "nap" is chosen. Comment other roles out. Check that "bridge" is selected as pan scheme (it is uncommented).
2. Restart PAN:

   ```
   btpan stop
   btpan init nap
   ```

   You need to restart PAN only this time. Next time you reboot your computer PAN is initialized automatically as NAP.

3. Edit /etc/network/interfaces file. First, let's set up the bridge configuration. Here you have two options. If your network connection is set up using DHCP add the following lines to the file. Replace *eth0* with your network interface name.

   Example 4-1. Bridge setup via DHCP

   ```
   auto br0
   iface br0 inet dhcp
     pre-up ifconfig eth0 down
     pre-up brctl addbr br0
     pre-up brctl addif br0 eth0
     pre-up ifconfig eth0 up
     post-down ifconfig eth0 down
     post-down brctl delif br0 eth0
   ```

   In case you have static IP address add following lines to /etc/network/interfaces. Remember to replace tags below with your network information. Also if your network interface card uses some other interface than *eth0* replace it too.

   Example 4-2. Bridge setup statically

   ```
   auto br0
   iface br0 inet static address <your_ip_address>
     netmask <your_net_mask>
     network <your_network>
     broadcast <your_network_broadcast_address>
     gateway <your_default_gateway>
     pre-up ifconfig eth0 down
     pre-up brctl addbr br0
     pre-up brctl addif br0 eth0
     pre-up ifconfig eth0 up
     post-down ifconfig eth0 down
     post-down brctl delif br0 eth0
   ```

Eth0 is now included in your bridge configuration. We need to add an another interface there too. This will be your Bluetooth device's interface. It is probably pan0. Add the following lines to /etc/network/interfaces

Example 4-3. Bluetooth device setup

```
auto pan0
iface pan0 inet manual
  pre-up brctl addbr br0
  pre-up brctl addif br0 $IFACE
  up ifconfig $IFACE 0.0.0.0 up
  down ifconfig $IFACE down
  post-down brctl delif br0 $IFACE
```

4. It is recommended to install *ifplugd* (on Debian). It will automatically bring your Bluetooth interface up when you plug the Bluetooth dongle or PCMCIA card into your computer. After installing *ifplugd* edit /etc/default/ifplugd file. Add there line:

   ```
   INTERFACES="pan0"
   ```

   When you are setting your system up you need to bring bridge up manually . Give command:

   ```
   ifup br0
   ```

   Next time your computer is rebooted bridge is set up automatically. If you did not install ifplugd utility you have to bring up also Bluetooth device interface pan0 manually. Give command:

   ```
   ifup pan0
   ```

Note! You do not need to enable IP forwarding or add rules to iptables when using bridge configuration. Next time when you will reboot the computer the bridge is set up automatically. You can now proceed setting the client computers (PANUs) up.

4.3. Setting up the Clients

From the client's point of view there is no difference whether your server has router or bridge configuration. The following steps are needed to establish a connection to a NAP.

1. When you are connecting the first time edit /etc/affix/affix.conf file. Check that value "pan_role="panu" is enabled and role "nap" is disabled. After editing configuration file restart PAN giving the following commands:

   ```
   btpan stop
   btpan init [panu]
   ```

2. Connect to NAP you have set up. Give command:

   ```
   btpan -s connect <address>
   ```

Parameter -s prevents SDP query as you probably do not have SDP daemon running on the NAP server. If you leave the parameter out remember to start SDP daemon on server (run **btsrv**).

To check your connection type:

```
cat /proc/net/affix/pan
```

This lists all your active PAN connection. The example below states that your host is PAN user and it has connection to NAP which Bluetooth device address is 00:0c:76:b1:08:76.

Example 4-4. Output of cat /proc/net/affix/pan

```
role: PAN User
no local protocol filter
local multicast filter:
  - 33:33:ff:29:02:79
  - 33:33:ff:29:02:79
  - 33:33:00:00:00:01
  - 33:33:00:00:00:01
connections:
  - 00:0c:76:b1:08:76
no protocol filter
no multicast filter
```

The previous phase was analogous to connecting network cable to a traditional network interface card. The physical connection now exists. To transfer data over the line you still need to set the IP address for the network interface (pan0). You can try to run DHCP client manually or just set the address using **ifconfig**. Do one of the following.

- If you had DHCP server running on the NAP or it was configured as a bridge and your LAN has DHCP server you use DHCP to obtain the IP address.

  Add setting for pan0 interface to /etc/network/interfaces file to use **ifup** and **ifdown** scripts to bring network connection up and down. Add the line below into the file. It is also recommended to install *ifplugd* (on Debian) utility. It will automatically bring pan0 interface up when you plug your Bluetooth device in your computer. The ifplugd configuration is as for router. See the previous section for instructions.

  Example 4-5. PANU pan0 interface up configuration

  ```
  iface pan0 inet dhcp //if you do not use dhcp replace ’dhcp’ with ’manual’
  ```

If you did not install *ifplugd* you can bring the network connection up easily giving **ifup pan0** as a root and respectively bring it down using command **ifdown pan0**.

To start DHCP manually give command **dhclient pan0**

- In case you do not have DHCP set the IP address manually. Give command **ifconfig pan0 <address> up**.

Now you have a network connection established. You can add more PANUs (up to 7) repeating these steps in each client computer.

* *Note! If you have problems accessing network check that the routing information is correct. It should use pan0 interface as a default gateway. If you give* ***route*** *command the routing information might look like the following.*

Example 4-6. Routing table

```
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
172.21.38.0 * 255.255.254.0 U 0 0 0 pan0
default herl1-cgw1.rese 0.0.0.0 UG 0 0 0 pan0
```

Chapter 5. RFCOMM Connections

RFCOMM emulates RS-232 serial line settings. RFCOMM provides multiple serial connections that are multiplexed on L2CAP to single connection. RFCOMM connections can be used for example together with some legacy applications that requires RS-232 connection.

5.1. LAN Connection

Although it is possible to establish a LAN connection over RFCOMM line it is recommended to use PAN connection for this (see the previous chapter). RFCOMM LAN connection is based on PPP. It is meant to be used with ordinary modems which have no error correction and thus it has lots of features that is not actually needed with Bluetooth devices. This results in unnecessary overhead that limits the connection bandwidth.

Currently Affix does not support encrypted PAN connections. If you want to use secure connection you have to use RFCOMM connection. This is going to be changed in the near future.

First you have to set your security setting. See chapter three for further information on this. Then start **btsrv** on the server computer where you connect from the client.

In the client computer give the following command to connect to the server:

```
btclt connect <addrd> LAN
```

Then start PPP as root with command:

```
pppd /dev/bty0
```

The port number (bty0) might vary. The port was echoed into the console when you connected to the server. To check that the connection is up and the attached port use command:

```
btclt status
```

Now you should have a network interface *ppp0* up. The default IP address for server is 192.168.0.1 and for the client 192.168.0.15. You can also configure your server as router or a bridge to allow connection to LAN from the remote computer. See instruction in the previous section.

Chapter 6. Affix And Other Bluetooth Equipments

This chapter gives some examples on how to connect a computer with other devices with Bluetooth support like mobile phones. These instruction was tested with following Nokia phones: N-Gage and 7650, but they should work with other Bluetooth capable mobile phones as well. Here is not explained how you should use your phone. You will find this information from equipment's manual. All the Bluetooth capable phones have almost the same functions. Only the menus might differ. This is why the phone usage is not covered here.

6.1. Transferring Files

Bluetooth supports two different ways of transferring files. First you can do it with interactive program like an ordinary FTP client. The another method is to push files from a Bluetooth device to another one. Both methods are described here.

Depending on your security settings probably pairing is needed. See the chapter three for instructions on this topic. To transfer files interactively start the file transfer client **btftp**.

- Start the file transfer program **btftp** in the directory you have the files to transfer.
- Enable Bluetooth support in your phone.
- Search for the devices around your computer. Give the *discovery* command.
- Now you can open the connection with command *open <your_mobile_phone_addr>*
- Accept the connection in your phone and you are ready to transfer files.
- Use command *put <filename>* to transfer files to the phone.
- To get files use *get <filename>* command.
- Finally close the connection with command *close* and exit the transfer program using command *quit*.

Pushing files is somewhat easier. It is also done with command **btftp**. Pushing differ in a way that all the previous steps are done at once. The connection is opened, the file is pushed to the remote device and the connection is closed. Use following command to push files into your mobile phone: **btftp push <phone_addr> <local_filename> [remote_filename]**.

6.2. Dial-up Connection Using Mobile Phone

Creating dial-up using Bluetooth connection and your mobile phone is almost the same as you would use ordinary phone and modem. There are two phases. Connecting to the phone (plug in the cable old days) and establishing the PPP connection over a serial line. You should start with configuring your PPP connection. You get the parameters from your ISP. Use pppconf (in Debian) to create a new PPP connection profile.

Now you need to create the Bluetooth connection between your computer and mobile phone:

```
btclt connect <phone_address> DUN
```

That will do it. DUN specifies the used connection profile and stands for Dial-Up Networking. Remember that pairing is needed also here (see the chapter three).

Now you can establish the connection. For example in Debian as a root give command:

```
pppd call <your_ppp_profile_name> /dev/bty0
```

PPP profile defines the parameters needed to connect your ISP. Note! Change the serial device '/dev/bty0' to correspond your device. Most probably it is the same. To find out the device give command:

```
btclt status
```

* *Note! If you have problems with PPP check that you have enabled the support for PPP protocol in your kernel.*

6.3. Using GPRS

Using GPRS is basically the same as analog dial-up connection. See previous section. You can use automatic script to establish the connection or make it manually. To connect with automatic script type command **btgprs** and press enter. It will guide you further.

You can set up the connection manually giving needed two commands. Change the device name below if needed. Format is /dev/bty<line_number>. To check the used line number type:

```
btclt status
```

Then connect to the phone and start PPP.

```
btclt connect <phone_address> DUN
pppd call gprs /dev/bty0 //gprs is a predefined GPRS profile installed by Affix
```

If you have problems connecting see /etc/ppp/peer/gprs-connect-chat file. It has the script that is used to connect to Internet. It works at least in Finland, but the needed script might be different depending on your location. Ask your ISP for further information.

6.4. Using Keyboard, Mouse or Other HID Equipments

Originally Human Interface Device (HID) defined a subclass for USB. It specifies different devices for user input. For example mouse, keyboard and other pointers. Bluetooth's HID profile was specified to get rid of the cable the different USB HID devices has. It allows wireless communication for USB HID devices as RFCOMM profile does for serial devices.

Starting from version 3.2.0 Affix supports HID profile. Affix has been tested with the following devices:

Table 6-1. HID device compatibility list

Device	Manufacturer
diNovo Media Desktop (keyboard, mouse, numerical pad)	Logitech
Optical Mouse MX 900	Logitech
Cordless presenter	Logitech
Wireless optical Keyboard	Microsoft
Wireless Optical Mouse	Microsoft

Using HID devices with Affix is fairly easy. The management is done completely with **bthidctl** utility. (This utility is covered in etail later in this documentation.) In the kernel *affix_hidp* module implements the HID support. This module is normally loaded into kernel automatically and you do not do anything to load it.

> **Note:** This is not true in Affix 3.2.0. affix_hidp module have to be loaded manually. Give command "**modprobe affix_hidp**" as root user.

When you are attaching a device first time to your computer you should let Affix first discover the devices for you. To do this press the connect button in your HID device or devices and give the following command as a root user:

```
bthidctl connect all
```

This command first search for Bluetooth devices around you and then checks if those are HID devices. If they are Affix tries to connect them. After successfully connecting you will see a message on the screen and you are able to use your device. Found device is also added to a HID device database which Affix maintains. Also ordinary users has access to this database and thus the HID device installation in the future is even simpler (e.g. After reboot).

Note, that Affix discovers only those HID devices which connect button was pressed. So if you neighbour happens to use such device it will not be discovered. Also if you use your HID device with two computers you have to always connect to the device like it was the first time after you change the computer. This is because HID devices remember the computer they were connected previously and wants to be connected with it.

To prepare all the devices stored in HID device database for usage give the following command (no need to be root).

```
bthidctl listen all
```

This prepares kernel for the devices. Now when a device tries to contact your computer the connection is established and the message is delivered to the kernel. This is when you move a HID mouse or press a button on a HID keyboard. Also, it might be good idea to put above command to some boot script so that your HID keyboard would work also after rebooting you computer.

All above commands had a parameter "all". You can replace it with your device’s Bluetooth address. Then only the specified device is connected or prepared. "all" refers to all the Bluetooth devices in your computer’s radio range. To see the status of connected devices type command:

```
bthidctl status
```

This gives you a list of all HID devices in the database. List contains device address, status (whether it is connected or not) and device’s type. Also /proc contains list of connected devices. To read this list give command:

```
cat /proc/net/affix/hid
```

To disconnect a HID device or all of them from your computer give command:

```
bthidctl disconnect all
```

Again, replace "all" with a specific device’s address if you want to disconnect only one device. After disconnecting a device it is not usable until you set kernel to listen it again. Of course you can also use connect method, but you probably do not want to do it as the device is already in the HID device database.

Chapter 7. Using Affix Utilities

This chapter introduces different utilities used to control Affix and your Bluetooth equipments. General format is that each utility has lot’s of command and it’s parameters. So the general format is:

```
utility <command> [<parameter1> <parameter2>...]
```

The following example uses **btclt** utility to perform *discovery* (command) and specifies that the discovery lasts 6 seconds as its first and only parameter.

```
btclt discovery 6
```

These instructions apply to Affix version 3.0.0.

I. btctl

btctl

Name

`btctl` — Control the Bluetooth device attached to your computer and the Affix operation.

Synopsis

btctl [-i <name>] [-a] <command> [parameters..]

Description

Manage your bluetooth device and Affix. Affix is a Bluetooth protocol stack for Linux. *Btctl* consistsof several sub commands. They are grouped according to their purpose and listed here below.

-i The interface name which settings you want to manage. E.g. "bt0"

-v Verbose mode.

-V Print version number.

-m <map_file> Set the uart map file.

-r Set the uart rate

General Commands

Here belongs commands for device initialization, debuging and to get help.

help <command name>

Show quick help for this command

debug [+|-][<module>|<detail>|all]

Allow or disallow debugging information for a certain module. Affix prints Debug information to system log. To get debug messages you can give the following command as a root user.

```
tail -f /var/log/syslog
```

Alternatively you can read the messages from /var/log/kern.log

initdev

Resets the device back to the initial state. Basically the same as *down* command below.

up

Set the Bluetooth network interface up. Works like *ifconfig* command.

down

Bring the Bluetooth network interface down. Works like *ifconfig* command.

capture <file> | -

Capture all traffic to a file. All the network packets that propagates through Affix are stored to the file given as a parameter. Giving ’-’ captures to standard output. Press ctrl-c to stop capturing. See Affix technical documentation for further information.

Security Commands

Use these commands to set pin codes for devices, create pairings and set your security level.

security <mode>

Set the used security level. *Mode* is on of the following: *open*, *link*, *service, pairable*, *auth*, *encrypt*.

addpin [<address>|default] <pin>

Add a pin code to pin code database. The pin code is linked to certain Bluetooth device address. If default keyword is used instead of a device address the pin code is applied to all addresses that are not listed in the pin code database.

rmpin [<address>|default]

Deletes a pin code that linked to given device address. Giving "default" keyword the default pin code will be deleted.

unpair <address>

Remote pairing with a remote device. Remote device’s address is given as a parameter.

pair <address>

Make pairing with a remote device. Depending on the security level you might need to set pin code first or you have to run *btsrv* which prompts you when the pin code is needed.

HCI Commands

Command’s in this group are used to control the Bluetooth device. These command operates at low level and an ordinary end user probably do not need these.

bdaddr

Print device’s Bluetooth address.

name [<name>]

Set the name for the device. This name is shown to others inquiring your device. If the name is not given as a parameter the current is printed.

scan [+|-][disc|conn]

Enables or disables scanning for connect or discovery attempt. If disabled this device cannot be connected or seen by remote devices. Enable discovery to allow others to see your device (*disc*) and to enable connecting (*conn*) to allow remote device establish a connection. Plus and minus signs in the front of keywords states enabled and disabled respectively.

remotename <bda>

Get the remote device’s name.

role <allow|deny> <master|slave>

Set the devices initial role to *master* or *slave*. Master device establishes the Bluetooth connection. First parameter defines if a role switch is possible. If allowed then this device can change its role according to the situation.

class [<class>]

The class of device parameter is used to indicate the capabilities of the local device to other devices. Give the class in hex decimal in form 0x1234.

auth <address>

Try to authenticate the connected device with given address. This command is used to try to authenticate the remote device associated with the specified Connection Handle. On an authentication failure, the Controller or Link Manager shall not automatically detach the link. Instead you can issue a disconnect command to terminate the link if the action is appropriate.

pkt_type [<0xXXX> | <mnemonic>]

This command is used to change packet types can be used for a connection that is currently established. This allows current connections to be dynamically modified to support different types of user data. Multiple packet types may be specified for the command parameter by bitwise OR operation of the different packet types.

page_to [to]

This parameter defines the maximum time the local Link Manager will wait for a baseband page response from the remote device at a locally initiated connection attempt. If this time expires and the remote device has not responded to the page at baseband level, the connection attempt will be considered to have failed.

hdisc <handle | address>

Close an ACL connection.

hconnect <address>

Create an ACL connection to remote device.

hpkt <address> [pkt_type]

Specify what packet types are allowed to transmit over a connection.

readbroad

Prints how many times broadcast packets are sent.

writebroad

Broadcast packets are not acknowledged and are unreliable. The number of broadcast retransmissions parameter is used to increase the reliability of a broadcast message by retransmitting the broadcast message multiple times. This parameter defines the number of times the device will retransmit a broadcast data packet.

reset

Reset the device.

ping <bda> <size>

Ping a remote device which BDA was given as a parameter. The ping packet size is defined as the second parameter.

lq <bda>

Print the link quality value. It ranges from 0-255, which represents the quality of the link between two Bluetooth devices. The higher the value, the better the link quality is.

rssi <bda>

Read the value for the difference between the measured Received Signal Strength Indication (RSSI) and the limits of the Golden Receive Power Range for a connection handle to another Bluetooth device.

The RSSI measurement compares the received signal power with two threshold levels, which define the Golden Receive Power Range. The lower threshold level corresponds to a received power between -56 dBm and 6 dB above the actual sensitivity of the receiver. The upper threshold level is 20 dB above the lower threshold level to an accuracy of +/- 6 dB.

cpl <bda>

Print current transmit power level.

linkinfo <bda> [<interval>]

Print all the above three link information.

pinq_start <length> [<max_period_len> <min_period_len> <len>]

This performs an automatic inquiry. Max_Period_Length and Min_Period_Length define the time range between two consecutive inquiries, from the beginning of an inquiry until the start of the next inquiry. The Controller will use this range to determine a new random time between two consecutive inquiries for each inquiry. The len parameter specifies the total duration of the InquiryMode and, when time expires, Inquiry will be halted.

pinq_stop <length>

Stop automatic inquiry

UART Commands

Use these command to manage serial Bluetooth devices.

init_uart <name> <vendor> [speed] [flags]

Used internally. You should use *open_uart* instead.

open_uart <name> <vendor> [speed] [flags]

Attach a serial Bluetooth device (for example iPaq users). This command maps the Bluetooth device to /dev/bty0.

close_uart <name>

Detach serial Bluetooth device.

AUDIO Commands

Currently SCP profile is supported only by PCMCIA cards. There are some problems using audio devices with USB dongles. The *bda* argument for the following command can be replaced by cache entry number (*btclt* list).

path 0x00 | 0x01

This is Ericsson specific command that sets the SCO data path. Setting 0x00 means that audio data is transmitted as usual using HCI. 0x01 means that audio is sent for example through PCM plug. This command needs hardware that supports it and has become nowadays almost obsolete.

audio on|off [sync | async] [setting <SCO setting>]

Configure audio settings. Option ’on’ enables audio support, the ’off’ disables it. Option ’async’ is used to enable the chip mechanism to signal affix to send packets to the chip. Option ’sync’ is used to enable affix to send synchrnously SCO packets to the chip. Option ’setting’ values can be found in section 6.12 page 387 in the BT Core V2.0 + EDR specification.

play <bda> [<file> | -]

Play a wav file. Audio data is in the given file sent to a Bluetooth headset. The alternative is to redirect standard input to headset. The following example directs raw audio data to a Bluetooth head phone.

Example 1. Audio play example

```
mpg123 -s <mp3_file_name> | sox -t raw -r 44100 -s -w -c 2 - -t raw -r 8000 -s -b -c 1 - | btctl play
<bda> -
```

record <bda> [<file> | -]

Record voice data from a microphone that supports Bluetooth. For example headset used with mobile phones.

II. btclt

btclt

Name

`btclt` — Get information about Bluetooth devices around you

Synopsis

btclt [-i <name>] <command> [parameters..]>

Description

Search for available Bluetooth devices. All the devices are stored to a device cache. To get the list of cached devices give *btclt* command without parameters. It will list cached devices, their names and cache entry number.

Later when you use other Affix commands you can use the cache entry number instead of writing the long Bluetooth device address. In general you can always replace the Bluetooth address with the cache entry number.

-i The interface name where your bluetooth device is attached. E.g. "bt0"

General Commands

help <command name>

Show quick help for the command given as a parameter.

list

Shows list of known devices (cached). This is default command and the result is the same if you run *btclt* without parameters.

flush

Clears the device cache.

HCI Commands

Host Controller Interface is used to manage the Bluetooth device. These command use the HCI to inquiry surrounding Bluetooth devices.

inquiry [length]

Search for surrounding Bluetooth devices. The parameter given is the time used to look devices up in seconds. Default is eight seconds. Found devices are cached.

discovery [length]

he same as above but tries also to sort out device names. Found devices are cached.

SDP Commands

Service discovery protocol (SDP) is used to find out services that a Bluetooth device can provide. SDP is sometimes also used to establishing the requested connection but this is invisible to the end user.

browse <address>

Sort out services on a remote Bluetooth device. This command returns you a list of commands that given remote device is able to provide you. Note that the address can be replaced with cache entity number.

search <address>

Almost the same as browse. Some devices does not support browsing. In this case use search method instead.

RFCOMM Commands

Serial ports can be used to transfer data. RFCOMM emulates RS-232 serial line settings. RFCOMM provides multiple serial connections that are multiplexed on L2CAP to single connection. RFCOMM connections can be used for example together with some legacy applications that requires RS-232 connection.

port <address> [channel | service]

Make a SDP request to get RFCOMM server channel for a service.

connect <address> [channel | service] [line number]

Connect to remote Bluetooth device. Channel number is mapped to device as /dev/bty<line>.

bind <address> [channel | service] [line number]

Bind a remote address to a given channel and line. Yet, connection is not established. When the channel is written the connection is established automatically.

disconnect [line]

Close the connection and unbind the channel.

status

Print the status of connections.

III. btpan

btpan

Name

`btpan` — Manage Personal Area Network (PAN) connection.

Synopsis

btpan [-i <name>] [-a] [--nospd|-s] <command> [parameters]

Description

Use this command to establish and manage Personal Area Network (PAN) connection. In a simple PAN called piconet is one network access point (NAP) and from one to seven PAN users (PANU). Piconets can be joined and this is called scatternet. Network access point might have also connection to LAN or Internet and it share this connection with PANUs acting as a gateway.

btpan accepts one option:

-s, --nospd Connect remote host directly without sending a SDP request to it first.

Commands

help <command name>

Show quick help for this command

init [role] [flags]

Initializes the host. A host has two roles: network access point and PAN user. Give this as a parameter. Keywords are "nap" and "panu". Role can be also defined in configuration file /etc/affix/affix.conf file. Edit the configuration file also to enable or disable automatic PAN setup on boot.

When PAN is initialized *ifconfig* command shows PAN interface named pan0 (if only one interface).

stop

Remove pan0 network interface and uninitialized PAN.

discovery [service]

Search for PAN hosts in your neighborhood. Service is one of the following keywords: NAP or PANU. SDP is used to discover hosts. To be discovered a host must have SDP server (*btsrv*) running.

connect <address> [service]

Establish a PAN. Address is the remote Bluetooth device’s address. If the remote host is not running Service Discovery Server (SDP) server use option -s to prevent SDP query, which would fail.

disconnect

Disconnect from a remote host.

ctl <interface> [m (start_addr stop_addr)*] [p (range_start range_stop)*]

Set protocol or multicast filter settings of your Bluetooth device. This allows the saving of the network bandwidth. After the interface name give whether ’m’ or ’p’ to add multicast or protocol filter respectively. This is followed by a set of accepted ranges. If no range is given the filter is reset to default.

For example if someone in the network is multicasting video. It is good idea to configure NAP so that video is not sent to Bluetooth devices to save the piconets bandwidth. See the examples below how to set filters.

You can find list of protocol types at

```
http://www.iana.org/assignments/ethernet-numbers
```

Multicast addresses for IPv4 are listed at

```
http://www.iana.org/assignments/multicast-addresses (IPv4)
http://www.iana.org/assignments/ipv6-multicast-addresses (IPv6)
http://www.rfc-archive.org/getrfc.php?rfc=2373 (IPv6)
```

Examples

```
// show list of all filters
btpan ctl pan0

// reset multicast filter
btpan ctl pan0 m

// Enable only IPv4 and ARP
btpan ctl pan0 p 0x800 0x800 0x806 0x806

// Enable only IPv6 neighbor discovery Multicast address range type
btpan ctl pan0 m 0x333300000000 0x333FFFFFFFF
```

IV. bthidctl

bthidctl

Name

`bthidctl` — Manage your Human Interface Devices (HID), like keyboards and mice.

Synopsis

bthidctl <command> [parameters..]

Description

Use this tool to control your Human Interface Devices (HID), like keyboards and mice. This utility communicates with kernel module and keeps database of HID devices. Database is stored in directory /var/spool/affix/hiddb.

Bthidctl can search for available devices and when found sends SDP request to them to find out if they are HID capable. All HID devices are added to device database. Note that you have to press device's connect button before *bthidctl* is able to find it.

Commands

help [command]

Print help for a command that was given as a parameter.

connect <bda>|"all"

Add devices in HID device database. Also reconnects to those devices that were connected to some other computer meantime. To connect a device it must be ready for it. Most devices have a special button that sets then ready for it. After connecting to device use "bthidctl listen" to connect again. Parameter is whether the connected device's Bluetooth address or keyword "all". All means that all connectible devices in the range are added to database and connected.

This command can be used only by root unless you change the user rights for the database directory at /var/spool/affix

listen [--active] <bda>|"all"

Notify kernel about device to allow HID connections to/from it. Before giving a device to listen command the device have to be registered to HID device database (see above). Give keyword "all" instead of Bluetooth address to activate all the devices included in the database. This is very useful for boot scripts.

Note! HID devices remember to which computer they last time were connected. If you used the HID device with some other computer meanwhile you have to use "bthidctl connect" to reconnect it to this computer.

--active Activate the connection immediately. This is needed only for some older HID devices that cannot automatically reconnect and can not announce it.

disconnect <bda>|"all"

Terminate a connection with given device and remove it from the kernel's allowed device list. The device database is left untouched. Device can be reconnected after adding it again to kernel's allowed device list ("bthidctl listen" command).

delete <bda>

Remove a device permanently from the device database. If there is open connection it will be closed. Note here is not possible use "all" keyword.

This command can be used only by root unless you change the user rights for the database directory at /var/spool/affix

status [discovery]

Get a list of HID devices in the device database. The given list includes device's Bluetooth address, status and device identifier. If "discovery" parameter is given a discovery is performed before listing the devices. The found devices will be listed too. HID incompatible devices are stated as "IN RANGE" status and they have no device identifier.

V. btftp

btftp

Name

`btftp` — Transfer files over Bluetooth connection.

Synopsis

btftp [-i <name>] [<command> [parameters..]

Description

Use this to transfer files over Bluetooth connection. FTP uses a client-server connection. To use FTP over Bluetooth run **btsrv** (see instruction below) in server host and btftp in client host.

You find download/upload directory on server at /var/spool/affix/Inbox. Locally files are stored and searched from the current working directory

General Commands

Start *btftp* without parameters. It will provide you with command prompt where you can give the following commands. You do not need to always write the whole device address as a parameter you can use the entry number instead. Use discovery and list commands to find out entry numbers. Note that entry numbers might change every time you do the discovery.

help <command name>

Show quick help for this command

list

Show the list of cached Bluetooth devices, their entry numbers, and addresses.

flush

Empty the list of cached Bluetooth devices.

inquiry [length]

Search for Bluetooth devices around you. Then length is the used search time in seconds.

discovery [length]

Search for Bluetooth devices around you and resolve their names. Then length is the used search time in seconds.

browse <address>

Browse services in some remote device

search <address>

Search for services in some remote device. Use this is browse results in an empty list. Some devices do not support browsing.

OBEX Commands

ftp

This is default when launching btftp. Shows command prompt where you can list and transfer files. This is like an ordinary console ftp client.

open <address> [<channel>]

Open a connection to a OBEX server with given address.

close

Close the connection with the server.

ls [<address> [<channel>]]

List the files at the remote server.

put [<address> [<channel>]] <local-file> [remote-file]

Transfer a file to remote server

get [<address> [<channel>]] <remote-file> [local-file]

Get a file from the remote server

push <address> [<channel>] <local-file> [remote-file]

Transfer a file to remote server. This open the connection for the transfer and closes after the transfer is done.

rm [<address> [<channel>]] <remote-file>

Delete a file from the remote server

cd <dir name>

Change the directory at the remote server.

mkdir <dir name>

Create a new directory at remote server.

VI. btsrv

btsrv

Name

`btsrv` — Run SDP server.

Synopsis

btsrv [-d] [-v] [--config config_file | -C config_file] [--expression config_expr | -e config_expr] [--initdev | -i] [--noinitdev] [--startsvc | -s] [--nostartsvc] [--managepin | -p] [--nomanagepin] [--managekey | -k] [--nomanagekey]

Description

Start services for other Bluetooth devices in your host. This should be run as root to allow access to Bluetooth devices.

To use graphical pin management btsrv should have access to display. You can grant this privilege with *xhost* command. For example "*xhost* +" allows all users access the display. Alternatively you can start the services also using *sudo*. When you have open X session give the following command in terminal: sudo *btsrv* Note! Before giving the command you have to add the user to sudoers list in /etc/sudoers file.

--nomanagepin Start *btsrv* without graphical user interface support. The pin code is not prompted when needed.

VII. btobex

btobex

Name

`btobex` — Run an OBEX server.

Synopsis

btobex -d <obex_root_dir>

Description

This allows clients to make ftp connections to this host. Use *btftp* command on the client host. *btobex* is used internally by *btsrv* command and you should run it instead of *btobex*.

-d Change the root directory where the transmitted files are stored. The default value is /var/spool/affix/Inbox.

VIII. btmodem

btmodem

Name

`btmodem` — Modem multiplexer and emulator utility.

Synopsis

btmodem [-e] [-b bluetooth] [-h] [-l [logfile]] [-m modem] [-q] [-v [-v]] [-x [count]]

Description

This reads data from a modem line and sends it to Bluetooth line and the other way around. If you have some legacy application that uses modem for transmitting data use *btmodem* to utilize Bluetooth instead of an ordinary modem. This is used mostly internally by other Affix utilities. This is mostly useful for the developers.

Parameters:

-e Modem emulation (use stdin/stdout). -m is not used

-b client The device name of the client serial connection, usually over a bluetooth connection. Default: /dev/bty0

-h Use hardware flow control (RTS/CTS) instead of the default XON/XOFF

-l [logfile] Log in a file. The default name for the logfile is /var/log/btmodem

-m modem The device name of the serial port on which the modem is connected. The default is: /dev/ttyS0

-q Quiet mode - do not show any messages on stdout. This has no influence on the information written in the logfile

-v Verbose mode: Show a lot of information as output on stdout and in the logfile if logging is activated. Usually only important status messages are shown.

Note! if both -q and -v are set, -q takes precedence!

-v -v Higher verbosity: Show Even more information, including all data transmitted in verbatim form.

-x [count] Show the data as hex dump (always, regardless of verbosity. If the quiet mode is set, don’t show the hex dump on the screen). The hex dump is written to the logfile. If -v -v and -x is set, the data will be shown twice, once verbatim and once as hex dump. Count is the number of bytes shown in each line. Default: 16

IX. btsdpd

btsdpd

Name

`btsdpd` — SDP server daemon.

Description

This is used only internally by Affix server which starts this when it receives a SDP request. Instead of using this you should use btsrv command.

X. btsocket

btsocket

Name

`btsocket` — Creates a client or a server socket.

Synopsis

For client socket:

btsocket [--l2cap] [--tty] [-2hbcfqrvw] [-i <name>] [-p command] bda <port | service>

For server socket:

btsocket [--l2cap] [--tty] [-2hbcfqrvw] [-i <name>] [-p command] -s [-l] port 1.

Description

Used internally by Affix utilities. Creates a client or a server socket.

Parameters:

-t Bind socket to tty (RFCOMM).

-2 Bind socket toL2CAP protocol. Use L2CAP to transmit datagrams.

-h Print usage and exit.

-f If set server forks on new connection.

-c If set sends carriage return after each buffer sent.

-w Socket is write only.

-p <command> Pipe data to this command.

-q Quit when the remote peer closes the connection.

-r Socket is read only.

-s Create a server socket.

-v Verbose mode.

-l Server remains in a loop.

-b Run yourself on background.

-W Wait forever although the connection was refused.