

Affix Bluetooth HOWTO

Andrei Emeltchenko

Nokia Research Center / Helsinki

andrei.emeltchenko at nokia dot com

Affix Bluetooth HOWTO

by Andrei Emeltchenko

Copyright © 2002, 2003, 2004 Andrei Emeltchenko

Revision History

Revision 0.1 May 30, 2002

Initial revision

Revision 2.0 July 12, 2003

Revision 2.1 2004

Table of Contents

1. General information	1
1.1. Copyright notice and disclaimer.....	1
1.2. Getting the latest version of Affix	1
1.3. Supported systems.....	1
1.4. Supported hardware.....	2
1.5. Mailing lists and other sources of information.....	2
1.6. Acknowledgements	2
2. Setting up Affix software	3
2.1. Prerequisites and kernel setup	3
2.2. Compilation and Installation	3
2.2.1. Configure, Compiling and Installing <code>affix-kernel</code> package	3
2.2.2. Configuring, compiling and installing <code>affix</code> package	4
2.2.3. Updating system configuration	5
3. Installing Debian Affix packages	7
3.1. Installing user space utilities	7
3.2. Building and Installing affix kernel modules	7
4. Connecting Bluetooth devices	8
4.1. PCMCIA Bluetooth devices	8
4.2. USB Bluetooth devices	8
4.3. Serial Bluetooth devices	8
4.4. Other Bluetooth devices	8
5. Usage and Features	10
5.1. Discovering bluetooth devices in the neighbourhood	10
5.2. Finding available services on the remote bluetooth devices	10
5.3. Connecting to the remote bluetooth device	11
5.3.1. Setting up PPP connection.....	11
5.4. Using OBEX for file transfer to/from mobile phone.....	12
5.5. Handling Phonebook and Calendar entries.	12
5.5.1. The format of the Phonebook entry	13
5.5.2. The format of the Calendar entry.....	13
5.6. Managing Profiles	13
5.7. Other tools shipping with affix package.....	14
5.8. Programming interface	15
5.8.1. User space API	15
5.8.2. Socket interface.....	15
5.8.3. Assigned numbers.....	16
5.9. Difference beetwen Bluez and Affix sockets	17
6. Affix GUI.....	19
7. FAQ.....	20
I. Reference Page.....	21
btctl.....	22

Chapter 1. General information

Affix - Bluetooth Protocol Stack for Linux has been developed at Nokia Research Center in Helsinki, Finland. Bluetooth is a new, wireless communication technology that provides short range connectivity for different appliances.

1.1. Copyright notice and disclaimer

Copyright (c) 2002, 2003 Andrei Emeltchenko

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts and with no Back-Cover Texts. A copy of the license is available at <http://www.gnu.org/copyleft/fdl.html> (<http://www.gnu.org/copyleft/fdl.html>).

This document may be included in commercial distributions without my prior consent. While it is not required, I would like to be informed of such usage.

This document is provided "AS IS", with no express or implied warranties. Use the information in this document at your own risk.

1.2. Getting the latest version of Affix

The latest version of Affix is available on the official Affix web site (<http://affix.sourceforge.net/#download>)

Now Affix is included in popular linux distribution Debian (testing/unstable) and you can apt-get all needed packages. Affix is divided into next packages.

```
affix - User space utilities for the Affix Bluetooth protocol stack.  
affix-common - Common files of the Affix Bluetooth protocol stack for Linux.  
affix-headers - Header files of the Affix Bluetooth protocol stack for Linux.  
affix-source - Driver source for the Affix Bluetooth protocol stack for Linux.  
libaffix-dev - Development files for the Affix Bluetooth protocol stack.  
libaffix2 - Libraries for the Affix Bluetooth protocol stack.
```

There are also rpm packages for RedHat like distributions. You can find them here (<http://affix.sourceforge.net/archive/RPM/>)

1.3. Supported systems

Affix Bluetooth software works on any Intel-based platforms, ARM-based platforms (iPaq, ARM9 and others), PowerPC-based platforms (iMac, IBM PowerPC STB). Package is fully dual-endian so in principle it would work on every linux platform.

1.4. Supported hardware

Latest Affix releases include drivers for all classes of Bluetooth devices such as:

USB based devices
PCMCIA based devices
UART based devices

For the full list of known supported devices see Affix Bluetooth supported device survey (<http://affix.sourceforge.net/hardware.shtml>)

1.5. Mailing lists and other sources of information

It is advisable to take a look at Affix page (<http://affix.sourceforge.net/>) for valuable information.

There are two Affix mailing lists:

Affix Developers' mailing list (<http://lists.sourceforge.net/lists/listinfo/affix-devel>) (Subscribe and Archives)

Affix Support mailing list (<http://lists.sourceforge.net/lists/listinfo/affix-support>) (Subscribe and Archives)

1.6. Acknowledgements

The writings from many people helped me to make this document possible. Many ideas came from different people. I just put them together. Special thanks to:

- Dmitri Kasatkin (<mailto:dmitri.kasatkin@nokia.com>)
- and many others who contributed to this work in one way or another.

Chapter 2. Setting up Affix software

2.1. Prerequisites and kernel setup

The following packages should be installed on your system:

```
libc - GNU C libraries and header files
pcmcia-cs - PCMCIA device manager for PCMCIA Bluetooth devices. This package is required if you use PCMCIA bluetooth a
libopenobex-<version>-dev - OBEX protocol development files.
libopenobex-<version> - OBEX protocol libraries.
```

If you want to have affix pin gtk window you need to install `python GTK` library.

Affix now supports `2.4.x` and `2.6.x` kernels. You need to have complete kernel tree for compiling because drivers contain reference to it.

It is also recommended to install `hotplug` package if you want to have hotplug USB capabilities.

2.2. Compilation and Installation

Now affix package is splitted into 2 separate packages: `affix` and `affix-kernel`

`affix-kernel-<version>` provides driver's source and headers for `affix`.

`affix-<version>` contains user-space utilities for affix bluetooth protocol stack. Note that you should compile and install `affix-kernel` before configuring `affix`.

I assume that you downloaded latest affix version as described in Section 1.2.

2.2.1. Configure, Compiling and Installing `affix-kernel` package

1. Configuring.

In order to configure affix you should execute

```
$ make config
```

in the Affix root directory and follow instructions. If you don't need debugging say No in the appropriate question.

2. Compiling.

To compile affix run:

```
$ make all
```

in the software root directory

3. Installing.

To install software make root and run:

```
# make install
```

in the `affix-kernel` root directory to install the driver's loadable modules into the system directories.

2.2.2. Configuring, compiling and installing `affix` package

1. Configuring.

In order to configure `affix` execute

```
$ ./configure
```

in the `affix` root directory. You can supply configuration script with optional parameters. Most important are shown below.

```
--enable-audio          use audio (default is "yes")
--enable-rfcomm         use rfcomm (default is "yes")
--enable-pan            use pan (default is "yes")
--enable-uart           use uart (default is "yes")
--enable-sdp            use sdp (default is "yes")
```

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

Others you can see by executing

```
$ ./configure --help
```

2. Compiling.

To compile affix run:

```
$ make all
```

in the software root directory

3. Installing.

To install software make root and run:

```
# make install
```

in the `affix` root directory to install the libraries, affix control programs, configuration files and documentation to the system directories.

2.2.3. Updating system configuration

If you have PCMCIA Bluetooth devices:

Basically Affix installation script copies all appropriate files to `/etc/pcmcia` and there is no need to edit anything. But in a case if you have old or broken PCMCIA package you can manually edit `/etc/pcmcia/config` file. For example for Nokia Connectivity Card:

1. Add next lines to the device driver definition:

```
device "btuart_cs"  
class "btuart_cs" module "btuart_cs"
```

2. Add next lines to the card definition before the serial devices (`serial.cs`)

```
card "Nokia Mobile Phones DTL-1"  
    manfid 0x0124, 0x1000  
    bind "btuart_cs"
```

If you have USB Bluetooth devices it is recommended to install `hotplug` package. Then you are able to plug in new device and use it immediately. The appropriate module `btusb.o` will load automatically if needed. Otherwise you need to load it manually.

Chapter 3. Installing Debian Affix packages

3.1. Installing user space utilities

To install user space utilities run:

```
# apt-get install affix
```

It will install also affix libraries.

3.2. Building and Installing affix kernel modules

First install module source:

```
# apt-get install affix-source
```

Now we are ready to compile modules. This includes following steps:

1. Make sure you have sources for kernel you are running.
2. Go to `/usr/src` and unpack the driver source archive:

```
$ cd /usr/src
$ tar xzvf affix.tar.gz
```

Now affix source is in `/usr/src/modules/affix`.

3. Go to your kernel source (i.e. `/usr/src/linux`) directory and run `make-kpkg`:

```
$ cd /usr/src/kernel-source-<version>
$ make-kpkg --added-modules affix modules_image --rootcmd fakeroot
```

4. Then you get package `affix-modules-<version>.deb`. Install it with:

```
# dpkg -i affix-modules-<version>.deb
```

Chapter 4. Connecting Bluetooth devices

4.1. PCMCIA Bluetooth devices

If you use latest pcmcia package installation script add configs for bluetooth device in `/etc/pcmcia` directory. You only need to insert your device to the pcmcia socket of your PC. PCMCIA card manager will load appropriate driver and bluetooth pcmcia device can be used for communication.

You also can virtually insert device using command

```
cardctl insert [slot number]
```

4.2. USB Bluetooth devices

After compiling and installing the Affix modules the hot-plug daemon will automatically load the `btusb.o` module if a compatible device is plugged into the USB socket.

4.3. Serial Bluetooth devices

In order to use serial bluetooth device you should load first `affix_usb` module. Then you need open uart with command

```
btctl open_uart {device name} [speed] [flags]
```

For example to open device connected to `/dev/ttyS0`:

```
$ btctl open_uart /dev/ttyS0 any 57600
```

4.4. Other Bluetooth devices

Bluetooth chips can be integrated to PDAs and PCs. In fact these integrated bluetooth devices are connected either to Serial line or USB. For example Compaq's iPaqs have bluetooth CSR chip connected to Serial line and

we can get it working by `btctl open_uartcommand`. For detailed information about Affix on iPaq see this (<http://affix.sourceforge.net/ipaq.shtml>) page.

Chapter 5. Usage and Features

5.1. Discovering bluetooth devices in the neighbourhood

There are two basic commands: `inquiry` and `discovery`. Both commands finds other bluetooth devices but `discovery` also shows name of the device but a bit slower.

Typical output of `discovery` command is shown below.

```
niko@niko:~$ btctl discovery 5
Searching ...
Searching done. Resolving names ...
done.
+1: Address: 00:02:ee:46:22:39, Class: 0x502204, Name: "Nokia 7650"
    Phone (Cellular) [Object Transfer,Telephony]
+2: Address: 00:e0:03:74:4e:3d, Class: 0x56010C, Name: "Windows 2k"
    Computer (Laptop) [Networking,Rendering,Object Transfer,Telephony]
```

Each device has its own number in the left column. This is the cache number for the given device. Later to access this device you can use this number as well as the bluetooth address. + means that device is in the cache and is discovered. - means that device is in the cache but is not discovered this time. See reference for more info.

5.2. Finding available services on the remote bluetooth devices

Each bluetooth device has some number of services available for use by another device. In order to find services on the remote device you need to have Service Discovery Protocol (SDP) support in the Affix, which is enabled by default now. Services are located in hierarchical form.

To see available services on remote bluetooth device there are two commands `browse` and `search`. Some broken hardware or wrong implementation of Bluetooth stack (like in MS Windows) do not support `browse`. `Search` command tries to find all known services instead. For more info see reference page.

Output of `browse` command for mobile phone Nokia 7650 is shown below.

```
niko@niko:~$ btctl browse 1
Connecting to host 00:02:ee:46:22:39 ...
=====
Service Name: Fax
-----
```

```

SvcRecHdl: 0x10000
Service Class ID List:
  "Fax" (0x1111)
  "Generic Telephony" (0x1204)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
  Port/Channel: 1
Profile Descriptor List:
  "Fax" (0x1111)
  Version: 0x0100
Browse Group List:
  "PublicBrowseGroup" (0x1002)
=====
Service Name: Dial-up Networking
-----
... skipped
=====
Service Name: Bluetooth Serial Port
-----
... skipped
=====
Service Name: OBEX Object Push
-----
... skipped
=====
Service Name: OBEX File Transfer
-----
... skipped

```

Note that several devices (like Nokia 6210) require to add pincode to be browsable. Most of devices allow you to browse without pincode.

5.3. Connecting to the remote bluetooth device

To utilize services on the remote bluetooth device it is required to enter pincode of that device.

```
$ niko@niko:~$ btctl addpin default "1234"
```

This command adds default PIN code which will be used to connect to all devices.

5.3.1. Setting up PPP connection

As Nokia 7560 support Dial-up Networking service we can connect to it using `connect` command. First I added my laptop to list paired devices of my phone despite it is not necessary. If you have running `btstpv` small gtk application will ask for passcode, otherwise passcode is handled by `addpin`. Now we can connect to DUN service.

```
niko@niko:~$ btctl connect 1 DUN
Connecting to host 00:02:ee:46:22:39 ...
Service found on channel 1
Connecting to channel 1 ...
Connected. Bound to line 0 [/dev/bty0].
```

Now connection established and phone behaves like a modem on `/dev/bty0`. You only need to change device name in your `ppp` configuration files from `/dev/ttySx` to `/dev/btyx`.

5.4. Using OBEX for file transfer to/from mobile phone

The Object Exchange protocol can best be described as binary HTTP. OBEX is optimised for ad-hoc wireless links and can be used to exchange all kind of objects like files, pictures, calendar entries (vCal) and business cards (vCard).

OBEX is builtin in devices like PDA's like the Palm Pilot, and mobile phones like Nokia ones.

So far Nokia 7650 support OBEX Object Push and OBEX File Transfer as it is seen from the service list we can send/receive media files (for example images made by digital camera). To receive files we need only run `btstpv` server. All objects are saved in `~/Inbox`. To send object to mobile use `push` command. Next we will see how to send image to phone.

```
niko@niko:~$ btctl push 1 niko.jpg
Transfer complete.
28998 bytes sent in 9.38 secs (3091.47 B/s)
```

For detailed information about use OBEX see `btctl` manual page.

5.5. Handling Phonebook and Calendar entries.

The format of Phonebook and Calendar entries is a text format. Entries can be send and receive through OBEX transfer.

5.5.1. The format of the Phonebook entry

The format is outlined below.

```
BEGIN:VCARD
VERSION:2.1
N:Surname;Firstname
TEL;VOICE;PREF:8036686
TEL;VOICE:+358718036686
TEL;VOICE:+358504836686
TEL;FAX:+358718036858
EMAIL:firstname.surname@nokia.com
TITLE:Research Engineer
NOTE:GSM
END:VCARD
```

When you send this entry to the phone it will ask to save contact to the phone memory.

5.5.2. The format of the Calendar entry

This is my own Calendar entry indicating that I have taekwondo training at 16:00 with alarm at 15:45, repeated weekly.

```
BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
UID:8
DESCRIPTION:Taekwondo
DTSTART:20030407T160000
DTEND:20030407T170000
X-EPOCHAGENDAENTRYTYPE:APPOINTMENT
AALARM;TYPE=X-EPOCSOUND:20030707T154500;;0;CalenAlarmSound
CLASS:PRIVATE
DCREATED:20030407T000000
RRULE:W1 MO 20030708T000000
LAST-MODIFIED:20030716T164700
END:VEVENT
END:VCALENDAR
```

5.6. Managing Profiles

Profiles are handled by the `btsrv` application. The `/etc/affix/services` configuration file has to be set up first. Each line of it specifies a service. The four fields of a line are the following:

Profile

profile the service is based on. Currently Serial Port, Dialup Networking, OBEX Object Push, OBEX File Transfer and LAN Access profiles are supported.

Flags

The only flags available are `socket` and `tty`. They are mutually exclusive and determine whether the command executed to handle an incoming request will use the RFCOMM socket interface or the virtual port device (`/dev/btyXX`) created for the connection.

Command

The command executed for an incoming request. The command will be started in a system shell (`/bin/sh`) with its standard input and output being the new socket or virtual port file depending on Flags. The standard error will be redirected to `/dev/null` if `btsrv` is running as a daemon otherwise it will remain the same as standard error of `btsrv`.

After creating the services file the `btsdp` program has to be started to create and handle the local SDP database. Finally the `btsrv` program has to be started which will listen for incoming requests and spawn the appropriate command to service the request.

Both `btsdp` and `btsrv` use the `syslog` to log messages with `LOG_DAEMON` facility if running as daemons. Otherwise they will use the standard error.

5.7. Other tools shipping with affix package

`/etc/affix/ppp_server`

A shell script to set up a PPP connection with a remote Bluetooth device.

`/etc/affix/modememu`

Modememu is a simple tool which can be used to emulate a modem to a remote Bluetooth device. It can be used to connect from a remote Bluetooth stack that supports only the dialup networking profile. The Compaq iPaq with the Socket Bluetooth card is an example where this program is useful. It can also be used to connect from a windows machine using direct cable connection over Bluetooth. The sample services file in `/etc/affix` gives an example how to use it.

`/etc/affix/masquerade`

Shell script to masquerade a PPP connection with a remote Bluetooth device, so that it can communicate through the local network if it's present. In kernel configuration the following options should be enabled:

```
Networking options/Network packet filtering (replaces ipchains)
Networking options/IP Netfilter Configuration/IP tables support
Networking options/IP Netfilter Configuration/Full NAT
```

Networking options/IP Netfilter Configuration/MASQUERADE target support

You also have to install iptables utilities.

5.8. Programming interface

5.8.1. User space API

API provides interface to the Bluetooth driver for the user space programs and includes:

- HCI functions which are used to manage Bluetooth devices.
- RFCOMM functions. Used to create and destroy RFCOMM connections.
- SDP functions to access local and remote SDP databases.

API is implemented as a set of shared libraries (e.g. `libaffix.so`) which are used by client programs.

For more details about implemented functions look at the `include/affix/btcore.h`.

For more details how to use interfaces look at implementation of `tctl.c`, `btsrv-main.c`.

5.8.2. Socket interface

Programming interface includes Berkeley socket interface for the user space programs. This software introduce new protocol family - `PF_AFFIX`, consisting of protocols used in the Bluetooth stack. To create the bluetooth socket the standard `socket` function is used:

```
fd = socket(PF_AFFIX, type, protocol);
```

Parameter `type` can be `SOCK_RAW` or `SOCK_STREAM`, `protocol` defines which protocol to use with the socket.

Protocol family uses new socket address structure:

```
struct sockaddr_affix {
    sa_family_t family;
    BD_ADDR    bda;
    uint16_t   port;
```

```

        BD_ADDR      local;
};

```

where `family` is `PF_AFFIX`, `bda` is Bluetooth device address (6 bytes) and `port` is PSM or `server_channel` value of the destination protocol/service, depending on the protocol.

Currently two protocols are used in the `PF_AFFIX` family:

BTPROTO_L2CAP

Socket with this protocol is used to create L2CAP connection to another device. Socket type is `SOCK_STREAM` here. This socket is used for example by the SDP server.

Before socket can be used it should be connected to the particular device, using `connect` system call:

```
err = connect(fd, sockaddr, addrlen);
```

`send()` and `recv()` syscalls is used to send and receive packets form the devices.

BTPROTO_RFCOMM

Socket with this protocol is used to create RFCOMM connection to another device. Socket type is `SOCK_STREAM` here. This socket can be used by SDP protocol.

Before socket can be used it should be connected to the particular device, using `connect` system call:

```
err = connect(fd, sockaddr, addrlen);
```

For more details how to use interfaces look at implementation of `btctl.c`, `btsrv-main.c`.

5.8.3. Assigned numbers

These numbers assigned for testing purpose and should be registered to prevent conflicts. Those numbers have been selected as if they were not used in the kernel.

1. Line discipline for the Bluetooth card (with serial interface).

```
#define N_NCC 14
```

2. Type of the Bluetooth network device interface and BLUETOOTH packet type

```
#define ARPHRD_BLUETOOTH 0x8000
#define ETH_P_BLUETOOTH 0x0027
```

3. Socket protocol family.

```
#define PF_AFFIX 27
#define AF_AFFIX PF_AFFIX
#define SOL_AFFIX 277
```

4. Virtual terminal major number.

```
#define BTY_MAJOR 60
```

5.9. Difference between Bluez and Affix sockets

The Bluez and Affix use very similar programming socket interface. In the example below I will show how to adopt program design to work with Bluez to the Affix stack. The information about adoption remote control program you can find here (<http://niko.spb.ru/linux/howto/affix/bemused/>).

```
/* Bluetooth address structures */
bdaddr_t bdaddr;
#ifdef AFFIX
BD_ADDR bdaddr;
#endif

/* Socket address structures */
#ifndef AFFIX
struct sockaddr_rc loc_addr;
#else
struct sockaddr_affix saddr;
#endif

/* Create socket */
#ifndef AFFIX
if( (s = socket(PF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM)) < 0 )
#else
if( (s = socket(PF_AFFIX, SOCK_STREAM, BTPROTO_RFCOMM)) < 0 )
```

```

#endif

/* Socket parameters */
#ifndef AFFIX
    loc_addr.rc_family = AF_BLUETOOTH;
    loc_addr.rc_bdaddr = bdaddr;
    loc_addr.rc_channel = uint8_t(channel);
#else
    saddr.family = AF_AFFIX;
    saddr.bda = bdaddr;
    saddr.port = channel;
#endif

/* Bind socket */
#ifndef AFFIX
    if( bind(s, (struct sockaddr *) &loc_addr, sizeof(loc_addr)) < 0 )
#else
    if( bind(s, (struct sockaddr *) &saddr, sizeof(saddr)) < 0 )
#endif

```

After creating socket we can use `read` and `write` calls. So the later development shouldn't differ.

Chapter 6. Affix GUI

Affix comes with a set of command-line utilities that provide the functionality an end-user needs. However the command line is not always what all people like. Therefore many functions were wrapped into the GUI and made easy to use from the graphical user interface.

Affix Frontend Environment (AFE) is a GTK-based application. GTK, which stands for GIMP ToolKit, is a library for creating graphical user interfaces for the X Window System. In order to compile AFE you need to install `libgtk` development package. You can download source code and read usage guides in the Affix Frontend Environment page (<http://affix.sourceforge.net/afe/>).

Chapter 7. FAQ

Q: Why Nokia mobile phones close connection to serial port profile?

A: The serial port profile in Nokia phones is configured the following way: after connection established from other device (i.e. PC) phone brakes connection and try to connect back to be master. As I understood this is the sonsequence of communication between phone and Nokia PC Suite.

I. Reference Page

btctl

Name

btctl — Manage Bluetooth devices

Synopsis

btctl [*command*] [*parameters*]

DESCRIPTION

Manage Bluetooth device

btctl name [*devicenumber*]

Set device name returned by discovery. When executed without argument shows current device name.

btctl class {*deviceclass*}

Set class of the device. Returned by inquiry or discovery command. Remote devices may search for devices (interested in) with certain class.

btctl scan {+ | -} {disc | conn}

Makes device discoverable or connectable or visa-versa.

btctl auth {on | off}

Enable/disable authentication on connection.

btctl connectrole {master | slave}

Set role in a connection. Useful when some devices do not support role switch. In that case run:

```
$ btctl connectrole master
```

Discover other Bluetooth devices in the neighborhood

btctl inquiry [*length*]

Search for the devices in the proximity for *length* seconds.

btctl discovery [*length*]

In the user point of view does the same but returns also devices names.

Finding available services on the remote bluetooth devices

In order to find services on the remote device you need to have Service Discovery Protocol (SDP) support in the Affix. Services are located in hierarchical form.

btctl browse {*device*}

Browse services on *device* starting from Root. *device* can be either an address or a number in a device list shown by discovery, inquiry or list commands.

btctl search {*device*}

Some broken hardware or wrong implementation of Bluetooth stack (like in MS Windows) do not support browse. Search command tries to find all known services instead.

Working with device information in the cache

When you first executed inquiry or discovery command `btctl` will save information about bluetooth devices in a cache. Next command handle this cache:

btctl list

List devices in the cache.

btctl flush

Flush the cache.

Connecting to the remote Bluetooth devices

btctl connect {*address*} [*channel* | *service_type*]

Makes RFCOMM connections to the device. You can connect to different channels which can be referred to as channel number or service type. Service types available are: SERIAL, DUN, LAN, HEADset (Serial if omitted). After connection have been established you get virtual serial line between your linux Bluetooth device and remote Bluetooth device. Ports `/dev/bty[0..99]` are available for use.

btctl disconnect {*line*}

Disconnect line number *line*.

btctl status

Show information about connected RFCOMM lines.

btctl addpin [*address* | *default*] {*pin*}

Adds PIN code to connect to the device which require authentication. For example:

```
$ btctl addpin 00:00:00:00:00:00 "1234"
```

Adds PIN "1234" to connect to the device with given address.

```
$ btctl addpin default "1234"
```

Set default PIN code. Used when no PIN code for the device is found.

btctl rmpin [*address* | default]

Remove PIN code for connection to the given device. When used without arguments removes all added PINs. For example:

```
$ btctl rmpin 00:00:00:00:00:00
```

OBEX commands

open {*address*} [*channel*]

Open OBEX FTP connection to the device.

close

Close OBEX FTP connection.

ls [*address* [*channel*]]

List current folder on the remote device.

get [*address* [*channel*]] {*file name*}

Get file from the remote device.

put [*address* [*channel*]] {*file name*}

Put file to the remote device.

rm [*address* [*channel*]] {*file name*}

Remove file on the remote device.

cd [*dir name*]

Change directory on the remote device.

mkdir [*dir name*]

Make directory on the remote device.

push {*address*} [*channel*] {*file name*}

Transfer file to the remote device using OBEX Object Push Profile.

```
$ btctl push 1 nokia.gif
```

Push image to the bluetooth mobile phone.

Arguments {*address*} and {*channel*} are not used with commands: *ls*, *get*, *put*, *rm* in FTP (prompt) mode.

Handle UARTs

These command are used to enable/disable serial bluetooth devices.

btctl open_uart {*device*} {*vendor*} [*speed*] [*flags*]

Open serial bluetooth device connected to *device* line. Vendor can be: *csr* *ericsson* *any*.

btctl close_uart {*device*}

Close serial bluetooth device connected to *device* line.

Authors

Manual page is written by Andrei Emeltchenko. Original author of affix bluetooth stack is Dmitry Kasatkin.