

# A Generic C/C++ API for Wireless Interfaces in GNU/Linux Systems

Volkan Yazıcı  
Özyeğin University  
Dept. of Electrical & Electronics Engineering  
*volkan.yazici@ozyegin.edu.tr*

WISERLAB Group Meeting, İstanbul, 2010

- 1 Intro.
  - Motivation
  - Literature
- 2 Contrib.
- 3 Status
  - Where Are We?
  - hostapd
  - Roadmap
- 4 Demo
  - Getters
  - Scanning
- 5 Doc.
- 6 Questions

# Motivation

- A generic C/C++ API for network interfaces.
  - Almost full coverage of wireless network configurations.
    - iwconfig
    - iwlist
    - wlanconfig
  - Generic network interface operations.
    - ifconfig
    - route
- A **library**, not a command line tool.
- A core component of the **Connectivity Brokerage**.

# Literature

- A majority of the previous works were basically parsing command (e.g. `iwlist`) outputs.
- Accessing command line tools within a program execution flow is dangerous and not convenient.
  - Command outputs are unreliable.
    - High possibility of unknown outputs (bugs).
    - Hard to be in sync. with every release.
    - Requires a significant amount of work to parse and extract certain tokens.

# Literature (Cont'd)

- Other libraries with similar goals:
  - `libiw` of Wireless Tools for Linux,
  - `libnm` of NetworkManager.
- Cons:
  - not designed for embedded environments (library dependencies),
  - function calls are blocking,
  - lacks `ifup/ifdown` and routing table functionalities,
  - no fine-grained access to the configurations,
  - doesn't provide an easy to use unified API.

# Contribution

- a **unified C/C++ library** with on par features compared to iwconfig, iwlist, wlanconfig, ifconfig, route tools.
- not yet another command wrapper, but a written from scratch library accessing network drivers through `ioctl()` **calls to the kernel** and **nl80211** netlink.
- **fine-grained access**, that is, just get/set a single property at a time.
- **non-blocking function calls**.
- code base is not bloated with 10 year old backward compatibility issues.
- fully **documented, clean code**; suitable for research & education purposes.

# Where Are We?

- **Finished!**

- Website & Docs.: <http://vy.github.com/wapi/>
  - Source Code: <http://www.github.com/vy/wapi/>
- Tested on various hardware. (Atheros, Broadcom, etc.)
- Fully documented code base. (Thanks Doxygen.)
- Now available as an **official OpenWrt package!**

# hostapd

- We can alter client side configurations?
- What about AP side?
- hostapd is the de-facto user space daemon,
- but provides restrictions in terms of access:
  - You are not allowed to externally alter the configurations of wireless devices that are allocated by hostapd.
  - Access to hostapd is allowed in two ways:
    - hostapd control interface (undocumented)
    - hostapd.conf (in machine-readable, mature format)



# Roadmap

- Get rid off libiw dependency.
  - Because of a race-condition bug in n180211, delayed for a while.

# Getters: Source Code

```
int ret;
double freq;
wapi_freq_flag_t freq_flag;
char essid[WAPI_ESSID_MAX_SIZE + 1];
wapi_essid_flag_t essid_flag;

...

/* freq */
ret = wapi_get_freq(sock, ifname, &freq, &freq_flag);
printf("freq: %g\n", freq);
printf("freq_flags: %s\n", wapi_freq_flags[freq_flag]);

/* essid */
ret = wapi_get_essid(sock, ifname, essid, &essid_flag);
printf("essid: %s\n", essid);
printf("essid_flag: %s\n", wapi_essid_flags[essid_flag]);

...
```

# Getters: Output

```
ip: 192.168.1.111
netmask: 255.255.255.0
freq: 2.412e+09
freq_flag: WAPI_FREQ_AUTO
chan: 1
freq: 2.412e+09
essid: ozu
essid_flag: WAPI_ESSID_ON
mode: WAPI_MODE_MANAGED
ap: 00:14:C1:34:CE:83
wireless.c:451:wapi_get_bitrate(): Bitrate is disabled.
wireless.c:546:wapi_get_txpower(): ioctl(SIOCGWTXPOW): Invalid argument
```

# Scanning: Source Code

```
int sleepdur = 1;
int sleeptries = 5;
wapi_list_t list;
wapi_scan_info_t *info;
int ret;

/* start scan */
ret = wapi_scan_init(sock, ifname);
printf("wapi_scan_init(): ret: %d\n", ret);

/* wait for completion */
do {
    sleep(sleepdur);
    ret = wapi_scan_stat(sock, ifname);
    printf("wapi_scan_stat(): ret: %d, sleeptries: %d\n", ret, sleeptries);
} while (--sleeptries > 0 && ret > 0);
if (ret < 0) return;

/* collect results */
bzero(&list, sizeof(wapi_list_t));
ret = wapi_scan_coll(sock, ifname, &list);
printf("wapi_scan_coll(): ret: %d\n", ret);

/* print found aps */
for (info = list.head.scan; info; info = info->next)
    printf(">> @%xl %s\n", (size_t) info, (info->has_essid ? info->essid : ""));
```

# Scanning: Output

```
wapi_scan_init(): ret: 0
wapi_scan_stat(): ret: 1, sleeptries: 5
wapi_scan_stat(): ret: 0, sleeptries: 4
wapi_scan_coll(): ret: 0
>> @8214ce8l OzuNet.Visitors
>> @8214c90l Wozunacad
>> @8214c38l OzuNet.Visitors
>> @8214be0l canbazoglu
>> @8214b88l NWii
>> @8214b30l wozunguest
>> @8214ad8l Wozunacad
>> @8214a80l ZQNET29
>> @8214a28l wozunvisitor
>> ...
```

# Doxygen

- All available public API (functions, variables, constants, structs, enums, etc.) is fully documented.
- Driver specific turnarounds are documented.
- All documentation is integrated within the source code itself as comment lines.
- Using Doxygen to produce HTML, LaTeX, RTF, etc. output from these comment lines.

WAPI


- Wireless API (WAPI)
- Modules
  - Network Interface Accessors
  - Common Data Structures
  - Utility Routines
  - Scanning
- Data Structures
  - Data Fields
- File List
- Directory Hierarchy
- Globals

Main PageModulesData StructuresFilesDirectories

## Wireless API (WAPI)

v1

WAPI provides an easy-to-use function set to configure wireless network interfaces on a GNU/Linux system. One can think WAPI as a lightweight C API for `iwconfig`, `wlanconfig` and `ifconfig` commands. (But it is not a thin wrapper for these command line tools.) It is designed to be used in wireless heterogeneous network research projects and supported by BWRC (Berkeley Wireless Research Center) and WISERLAB (Wireless Information Systems Engineering Research Laboratory at Özyeğin University).

Generated on Wed Nov 10 2010 18:17:43 for WAPI by  1.7.1

WAPI

- Wireless API (WAPI)
- Modules
  - Network Interface Accessors
    - Frequency Accessors
    - ESSID (Extended Service Set)
    - Operating Mode
    - Access Point
    - Bit Rate
    - Transmit Power
  - Common Data Structures
  - Utility Routines
  - Scanning
- Data Structures
- Data Fields
- File List
- Directory Hierarchy
- Globals

Main Page
Modules
Data Structures
Files
Directories

Functions

## Scanning

This group consists of functions for scanning accessible access points (APs) in the range. [More...](#)

### Functions

int <b>wapi_scan_init</b> (int sock, const char *ifname)	Starts a scan on the given interface.
int <b>wapi_scan_stat</b> (int sock, const char *ifname)	Checks the status of the scan process.
int <b>wapi_scan_coll</b> (int sock, const char *ifname, <a href="#">wapi_list_t</a> *aps)	Collects the results of a scan process.

### Detailed Description

This group consists of functions for scanning accessible access points (APs) in the range.

Unfortunately provided scanning API by wireless-tools libraries (libiw) is quite limited, and doesn't list all APs in the range. (See `iw_process_scan()` of libiw. For `iwlist` case, it has its own hardcoded magic for this stuff and it is not provided by libiw.) For this purpose, we needed to implement our own scanning routines. Furthermore, scanning requires extracting binary results returned from kernel over a char buffer, hence it causes dozens of hairy binary compatibility issues. Luckily, libiw provide an API method to cope with this: `iw_extract_event_stream()`. That's the only place in this project relying on libiw.

The scanning operation disables normal network traffic, and therefore you should not



WAPI

- Wireless API (WAPI)
- Modules
  - Network Interface Accessors
    - Frequency Accessors
    - ESSID (Extended Service Set)
    - Operating Mode
    - Access Point
    - Bit Rate
    - Transmit Power
  - Common Data Structures
  - Utility Routines
  - Scanning
- Data Structures
- Data Fields
- File List
- Directory Hierarchy
- Globals

Search

Main Page   Modules   Data Structures   Files   Directories

## Access Point

### [Network Interface Accessors]

#### Functions

void <b>wapi_make_broad_ether</b> (struct sockaddr *sa) Creates an ethernet broadcast address.
void <b>wapi_make_null_ether</b> (struct sockaddr *sa) Creates an ethernet NULL address.
int <b>wapi_get_ap</b> (int sock, const char *ifname, struct sockaddr *ap) Gets access point address of the device.
int <b>wapi_set_ap</b> (int sock, const char *ifname, const struct sockaddr *ap) Sets access point address of the device.

#### Function Documentation

```
int wapi_get_ap ( int          sock,
                  const char * ifname,
                  struct sockaddr * ap
                )
```

Gets access point address of the device.

For "any", a broadcast ethernet address; for "off", a null ethernet address is returned. Returned struct sockaddr is of AF\_INET, AF\_INET6 family.

WAPI

- Wireless API (WAPI)
  - Modules
    - Network Interface Accessors
      - Frequency Accessors
      - ESSID (Extended Service Set)
      - Operating Mode
      - Access Point
      - Bit Rate
      - Transmit Power
    - Common Data Structures
    - Utility Routines
    - Scanning
  - Data Structures
  - Data Fields
  - File List
  - Directory Hierarchy
  - Globals

/home/vy/ozu/.../group\_txpower.html

```

00483     return ret;
00484 }
00485
00486
00487 int
00488 wapi_set_txpower(
00489     int sock,
00490     const char *ifname,
00491     int power,
00492     wapi_txpower_flag_t flag)
00493 {
00494     struct iwreq wrq;
00495
00496     /* Construct the request. */
00497     wrq.u.txpower.value = power;
00498     switch (flag)
00499     {
00500     case WAPI_TXPOWER_DBM:
00501         wrq.u.txpower.flags = IW_TXPWR_DBM;
00502         break;
00503     case WAPI_TXPOWER_MWATT:
00504         wrq.u.txpower.flags = IW_TXPWR_MWATT;
00505         break;
00506     case WAPI_TXPOWER_RELATIVE:
00507         wrq.u.txpower.flags = IW_TXPWR_RELATIVE;
00508         break;
00509     }
00510
00511     /* Issue the set command. */
00512     return wapi_ioctl(sock, ifname, SIOCSIWTPW, &wrq);
00513 }
00514
00515

```

# Questions?



answers of questions

I'm Feeling Lucky