Wireless
Wi-Fi

Dual Channel Wi-Fi Generic Linux Station Integration and Operations Guide

WR-GL-DCW-LINUX-STA-GEN-V01-190513

RELEASED

DISCLAIMER

This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein. Any use or reliance on the information or opinion in this document is at the risk of the user, and CableLabs and its members shall not be liable for any damage or injury incurred by any person arising out of the completeness, accuracy, or utility of any information or opinion contained in the document.

CableLabs reserves the right to revise this document for any reason including, but not limited to, changes in laws, regulations, or standards promulgated by various entities, technology advances, or changes in equipment design, manufacturing techniques, or operating procedures described, or referred to, herein.

This document is not to be construed to suggest that any company modify or change any of its products or procedures, nor does this document represent a commitment by CableLabs or any of its members to purchase any product whether or not it meets the characteristics described in the document. Unless granted in a separate written agreement from CableLabs, nothing contained herein shall be construed to confer any license or right to any intellectual property. This document is not to be construed as an endorsement of any product or company or as the adoption or promulgation of any guidelines, standards, or recommendations.

Document Status Sheet

Document Control Number:	WR-GL-DCW-LINUX-STA-GEN-V01-190513			
Document Title:	Dual Channel Wi-Fi Generic Linux Station Integration and Operations Guide			
Revision History:	D01 – Released 01/18/19 V01 – Released 05/13/19			
Date:	May 13, 2019			
Status:	~~Work in Progress~~	~~Draft~~	Released	~~Closed~~
Distribution Restrictions:	~~Author Only~~	~~CL/Member~~	~~CL/ Member/ Vendor~~	Public

Contents

1 SCOPE 5
1.1 Introduction and Overview 5
1.2 Purpose of Document 5
2 REFERENCES 5
2.1 Informative References 5
3 TERMS AND DEFINITIONS 5
4 ABBREVIATIONS AND ACRONYMS 5
5 ARCHITECTURE OVERVIEW 7
5.1 Component Responsibility Breakdown 7
6 HARDWARE COMPONENTS 7
7 SOFTWARE COMPONENTS 8
7.1 Individual Components 8
7.1.1 LIBDCWPROTO 8
7.1.2 LIBDCWSOCKET 8
7.1.3 DCSTAD 8
7.1.4 DCSTAD SCRIPT 8
7.2 Code Repositories 10
7.3 Building 10
7.3.1 BUILD MACHINE PREPARATION 10
7.3.2 BUILDING THE STATION DAEMON 10
8 PORTING TO OTHER PLATFORMS 11
9 USER GUIDE 11
9.1 Starting Dual Channel Wi-Fi with One Data Channel 11
9.2 Starting Dual Channel Wi-Fi with Two Data Channels 11
10 TROUBLESHOOTING 12
10.1 Validating Wi-Fi Adapter Connection States 12
10.2 Validating Interface Byte/Packet Counters 12
10.3 Linux Firewall 12
10.4 Reverse Path Filtering 12
10.5 Wireshark 12
10.5.1 DUAL CHANNEL WI-FI WIRESHARK DISSECTORS 12
10.5.2 BUILDING A DUAL CHANNEL WI-FI PROTOCOL-AWARE WIRESHARK 12
10.5.3 RUNNING THE DUAL CHANNEL WI-FI PROTOCOL-AWARE WIRESHARK 13
11 KNOWN ISSUES AND LIMITATIONS 13
11.1 Network Manager Conflicts 13

Figures

Figure 1 – Linux DCW Station S/W Architecture Overview 7

1 SCOPE

1.1 Introduction and Overview

This document describes how to build and use the Dual Channel Wi-Fi (DCW) feature on devices running the Linux operating system.

1.2 Purpose of Document

The purpose of this document is to explain the software compilation process for the Dual Channel Wi-Fi feature on Linux. In addition to the integration steps, a basic user manual and troubleshooting guide are included.

2 REFERENCES

2.1 Informative References

None

3 TERMS AND DEFINITIONS

This document uses the following terms.

data channel	A downstream-only Wi-Fi connection used in Dual Channel Wi-Fi for offloading traffic from the primary channel connection.
Linux	Open-source operating system created by Linus Torvalds.
OSX	Proprietary operating system developed and owned by Apple.
POSIX	Portable Operating System Interface—a set of APIs and command-line utilities to help standardize across different flavors of UNIX and UNIX-like systems.
primary channel	The main Wi-Fi connection used in Dual Channel Wi-Fi for both DCW signaling and upstream and downstream traffic.
Wi-Fi	A technology enabling the wireless transmission and reception of LAN traffic.

4 ABBREVIATIONS AND ACRONYMS

This document uses the following abbreviations.

AP	access point
API	application programming interface
DCW	Dual Channel Wi-Fi
IP	Internet Protocol
LAN	local area network
MAC	media access control

PSK	pre-shared key
SSID	service set identifier
WPA	Wi-Fi Protected Access

5 ARCHITECTURE OVERVIEW

Because of the variety of Linux distributions with various network managers and configurations, some integration effort may be required depending on the target deployment distribution. A more generic software architecture approach has been taken to assist with platform portability and to simplify integration efforts.

The current code release includes an example DCSTAD script that enables the Dual Channel Wi-Fi daemon to control data channel interfaces by using “iwconfig” and “wpa_supplicant” commands (Figure 1).

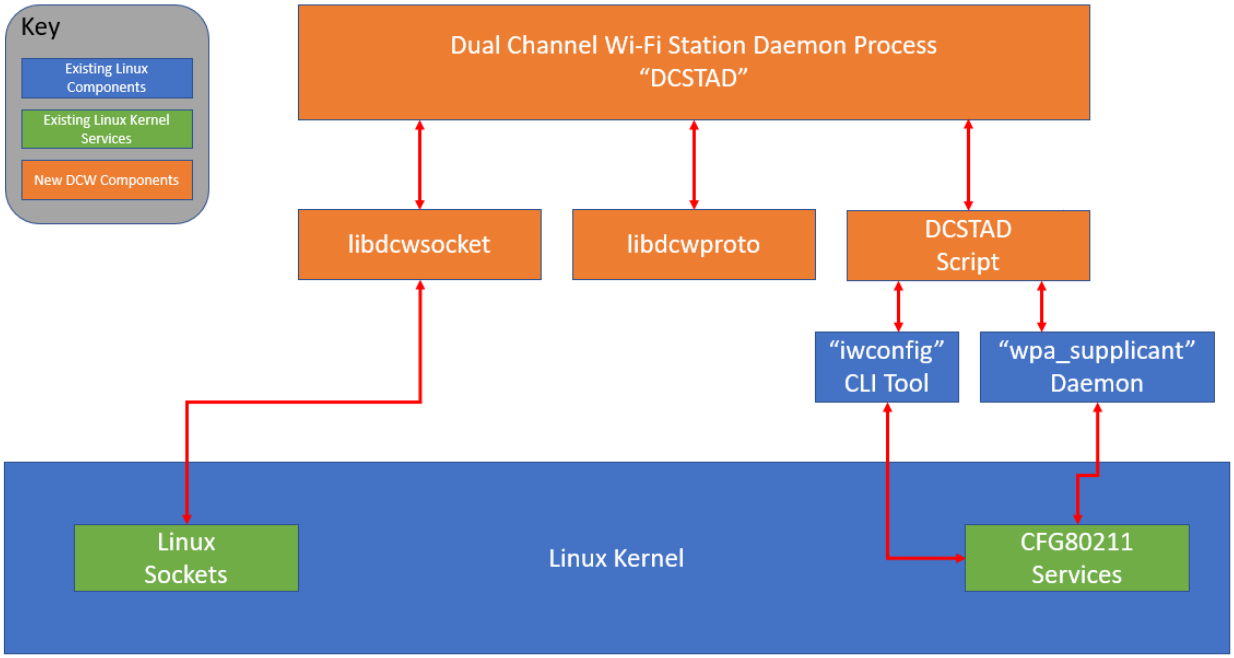

Figure 1 – Linux DCW Station S/W Architecture Overview

5.1 Component Responsibility Breakdown

The responsibility of each software component is separated as follows.

- DCSTAD—Controls all the logic and manages Dual Channel Wi-Fi. This component is not platform specific.
- DCSTAD Script—Implements hardware-specific details for setting up and managing the data channel Wi-Fi interfaces on the platform for which it resides.
- libdcwsocket—Implements platform-specific details for how to transmit and receive DCW signaling frames.
- libdcwproto—Implements DCW protocol for establishing connections with the access point (AP).

6 HARDWARE COMPONENTS

The following hardware components are required for building and running this software.

- A Linux PC or embedded device
- Two or more Wi-Fi adapters supporting the standards desired for use

7 SOFTWARE COMPONENTS

7.1 Individual Components

Dual Channel Wi-Fi station functionality comprises several software components.

7.1.1 libdcwproto

The *libdcwproto* component is a platform-independent C library responsible for marshalling and serializing the Dual Channel Wi-Fi signaling messages. The library models every Dual Channel Wi-Fi signaling message as a C struct and provides conversion to/from a raw byte-buffer, ready for transmission/reception.

This component is usable for both AP and station code.

7.1.2 libdcwsocket

The *libdcwsocket* component is a Linux- and OSX-specific C library that simplifies transmission and reception of Ethernet frames by using the CableLabs Ethertype code of 0xB4E3 and the CL3 protocol type of 0x00DC. More information regarding the specific details of the protocol can be found in the protocol specification document.

This component is usable for both AP and station code.

7.1.3 DCSTAD

The *DCSTAD* daemon component is the heart of the Dual Channel Wi-Fi business-logic implementation. The component handles all network protocol signaling for Dual Channel Wi-Fi. A "DCSTAD script" is required to grant this component the ability to configure the data channel Wi-Fi interfaces.

7.1.4 DCSTAD Script

The *DCSTAD Script* acts like a hardware-abstraction layer to the "DCSTAD" component. The script is called upon by the station daemon (DCSTAD) only when there is an event. The DCSTAD component itself has no knowledge of how to set up a service set identifier (SSID) on a specific adapter, but it does know which adapter needs to be joined to which SSID. For example, when the DCSTAD component has discovered a new AP capable of Dual Channel Wi-Fi and has received the parameters for the new data channel(s), it invokes the DCSTAD script to perform the actual join (channel-bond) operation of the data channel.

The DCSTAD script is responsible for the following:

- setting up the data channel Wi-Fi adapter parameters such as SSID, WPA, etc.;
- ensuring that traffic received on the data channel is routed into the system correctly;
- reporting back to the DCSTAD process with the result of the operation—success or failure; and
- restores the interface configuration to the original state when requested.

7.1.4.1 Example DCSTAD Script "dcstad-iwconfig.sh"

An example DCSTAD script can be found in the DCSTAD repository under the scripts/ directory. It is a Bash script that uses the "iwconfig" and "wpa_supplicant" command-line tools to provision the data channel interfaces.

7.1.4.2 DCSTAD Script API

The application programming interface (API) for this script is maintained in the DCSTAD (https://github.com/cablelabs/dcstad) repository as "dcstad-script-api.txt". Below is a reference version of the script API.

```
The "DCSTAD Script" is responsible for executing on the following events (reasons):
  . Join response from AP   -- "JOIN"
  . Unjoin ACK from AP      -- "UNJOIN"
  . DCSTAD Process Startup  -- "STARTUP"
  . DCSTAD Process Shutdown -- "SHUTDOWN"
```

7.1.4.2.1 AP Join Response (JOIN)

```
  The script is invoked as such:
    $ /path/to/dcstad-script
```

```
  Environment Variables:
    REASON                  -- "JOIN"
    REPLY_FD                -- The FD# for which to reply back to the daemon on.
    PRIMARY_INTF            -- The interface name used for the WiFi primary channel.
    PRIMARY_INTF_MACADDR    -- The MAC address of the primary channel interface.
    DATACHAN_INTF_COUNT     -- The count of data channel interface names used for this channel
bond. (must be >=1)
    DATACHAN_INTF_N         -- The interface name used for a data channel (example:
DATACHAN_INTF_0=wlan1, DATACHAN_INTF_1=wlan2, ...)
    DATACHAN_INTF_N_MACADDR  -- The MAC address of a data channel.
    DATACHAN_SSID_COUNT     -- The count of data channel SSID names used for this channel bond.
(must be >=1)
    DATACHAN_SSID_N         -- The SSID name available to be used used for a data channel bond
(example: DATACHAN_SSID_0=some_dcw_datachan1, DATACHAN_SSID_1=some_dcw_datachan2, ...)

  Exit Codes:
     = 0 -- Success (DCSTAD will ACK Server)
    >= 1 -- Failure (DCSTAD will NACK Server)

  Reply:
    For the "JOIN" reason, the script must tell the daemon which interfaces were
    joined to which SSIDs. The "REPLY_FD" environment variable is provided for
    this purpose. The file descriptor number in which to write the reply is
    provided in this variable.

    Example, if there is only one data channel interface ('wlan1') and one data
    channel SSID ('dcssid'), then the response to the "REPLY_FD" would be:
      "wlan1 dcssid"
```

7.1.4.2.2 AP Unjoin (UNJOIN)

```
  The script is invoked as such:
    $ /path/to/dcstad-script

  Environment Variables:
    REASON                  -- "JOIN"
    PRIMARY_INTF            -- The interface name used for the WiFi primary channel.
    PRIMARY_INTF_MACADDR    -- The MAC address of the primary channel interface.
    DATACHAN_INTF_COUNT     -- The count of data channel interface names used for this channel
bond. (must be >=1)
    DATACHAN_INTF_N         -- The interface name used for a data channel (example:
DATACHAN_INTF_0=wlan1, DATACHAN_INTF_1=wlan2, ...)
    DATACHAN_INTF_N_MACADDR  -- The MAC address of a data channel.

  Exit Codes:
     = 0 -- Success
    >= 1 -- Failure (not sure what to do here... log an error i guess?)
```

7.1.4.2.3 Daemon Startup (STARTUP)

```
  The script is invoked as such:
    $ /path/to/dcstad-script

  Environment Variables:
    REASON              -- "STARTUP"
    PRIMARY_INTF        -- The interface name used for the WiFi primary channel.
    DATACHAN_INTF_COUNT  -- The count of data channel interface names used for this channel bond.
(must be >=1)
    DATACHAN_INTF_N     -- The interface name used for a data channel (example:
DATACHAN_INTF_0=wlan1, DATACHAN_INTF_1=wlan2, ...)

  Exit Codes:
     = 0 -- Success (DCSTAD daemon will continue to startup and run)
    >= 1 -- Failure (DCSTAD daemon will abort and exit)
```

7.1.4.2.4 Daemon Shutdown (SHUTDOWN)

```
  The script is invoked as such:
    $ /path/to/dcstad-script
```

```
  Environment Variables:
    REASON               -- "SHUTDOWN"
    PRIMARY_INTF         -- The interface name used for the WiFi primary channel.
    DATACHAN_INTF_COUNT  -- The count of data channel interface names used for this channel bond.
(must be >=1)
    DATACHAN_INTF_N      -- The interface name used for a data channel (example:
DATACHAN_INTF_0=wlan1, DATACHAN_INTF_1=wlan2, ...)

  Exit Codes:
     = 0 -- Success (DCSTAD will continue to shut down)
    >= 1 -- Failure (DCSTAD will log an error and continue to shut down)
```

7.2 Code Repositories

All Dual Channel Wi-Fi code is stored in the CableLabs GitHub team "DCW," located at https://github.com/orgs/cablelabs/teams/dcw/repositories.

Each individual software component can be found in their respective git repositories.

Wireshark Dissectors	https://github.com/cablelabs/dcwwireshark
libdcwproto	https://github.com/cablelabs/libdcwproto
libdcwsocket	https://github.com/cablelabs/libdcwsocket
DCSTAD	https://github.com/cablelabs/dcstad

7.3 Building

These are the build instructions for compiling the station daemon process for the local host.

7.3.1 Build Machine Preparation

These instructions are dependent on starting on a freshly installed Ubuntu 14.04 64-bit server install.

```
$ sudo apt-get update
$ sudo apt-get install build-essential git automake
```

WARNING: SSH git repo keys may need to be set up if starting fresh.

7.3.2 Building the Station Daemon

This example demonstrates how to build the station daemon for the local machine, statically linking in all Dual Channel Wi-Fi dependency libraries. However, if desired, the libraries could be built and linked as shared objects.

First, clone all the code.

```
$ git clone git@github.com:cablelabs/dcstad.git
$ cd dcstad/
$ git clone git@github.com:cablelabs/libdcwproto.git
$ git clone git@github.com:cablelabs/libdcwsocket.git
```

Build the libdcwproto dependency as a static library.

```
$ cd libdcwproto/
$ ./configure --enable-shared=no --enable-static=yes
$ make
$ cd ..
```

Build the libdcwsocket dependency as a static library.

```
$ cd libdcwsocket/
$ ./configure --enable-shared=no --enable-static=yes
$ make
$ cd ..
```

Build the “dcwapd” daemon process.

```
$ CPPFLAGS="-I`pwd`/libdcwproto/include -I`pwd`/libdcwsocket/include" LDFLAGS="-
L`pwd`/libdcwsocket/src/.libs -L`pwd`/libdcwproto/src/.libs" ./configure
$ make
```

The output binary will be `src/dcstad` .

8 PORTING TO OTHER PLATFORMS

The current code has been tested on Ubuntu 14.04 64-bit Server and Raspbian. However, minimal or even possibly no porting effort may be required to run Dual Channel Wi-Fi on other CFG80211 Linux distributions. Linux distributions that use Wi-Fi services other than CFG80211 may require a new DCSTAD script. No other components should require modification.

9 USER GUIDE

9.1 Starting Dual Channel Wi-Fi with One Data Channel

For this example, the interface configuration will be used as follows.

- DCW AP primary channel SSID “DCWPrim”
- DCW AP WPA2 PSK password “ABCDEFGH”
- wlan0—primary channel Wi-Fi interface
- wlan1—data channel Wi-Fi interface

The user will have to manually set up the primary channel interface. Ideally, this task is performed by a network manager. For the sake of completeness, all steps including the initial connection to the primary channel interface are included. Ensure the network manager (if any) will not interfere with both the “wlan0” and “wlan1” interfaces.

Create the primary channel Wi-Fi configuration. Below is an example “/tmp/pc_wpa_supplicant.conf” file.

```
network={
        ssid="DCWPrim"
        scan_ssid=1
        psk="ABCDEFGH"
        key_mgmt=WPA-PSK
        auth_alg=OPEN
}
```

Bring up the primary channel interface.

```
$ ifconfig wlan0 0.0.0.0 up
$ iwconfig wlan0 mode managed
$ iwconfig wlan0 essid off
$ wpa_supplicant -B -c/tmp/pc_wpa_supplicant.conf -iwlan0
```

Start the Dual Channel Wi-Fi station daemon.

```
$ sudo src/dcstad -p wlan0 -d wlan1 -s scripts/dcstad-iwconfig.sh
```

9.2 Starting Dual Channel Wi-Fi with Two Data Channels

This example assumes the interface configuration is as follows.

- wlan0—primary channel Wi-Fi interface
- wlan1—data channel Wi-Fi interface
- wlan2—data channel Wi-Fi interface

Change the call to start the station daemon, adding another data channel (-d) interface parameter.

```
$ sudo src/dcstad -p wlan0 -d wlan1 -d wlan2 -s scripts/dcstad-iwconfig.sh
```

10 TROUBLESHOOTING

10.1 Validating Wi-Fi Adapter Connection States

The Wi-Fi adapter states can be viewed by running the “iwconfig” command.

```
$ iwconfig
```

This command will provide information about which SSID is joined to which adapter.

10.2 Validating Interface Byte/Packet Counters

Depending on platform availability, the interface byte and packet counters can be displayed through either of the following commands.

```
$ ifconfig -a
```

-- OR --

```
$ netstat -ian
```

10.3 Linux Firewall

In the event there are issues with traffic being arbitrarily filtered, it is a good idea to check the Linux “iptables” firewall. On most platforms, the entire running iptables firewall configuration can be dumped by executing the following command.

```
$ iptables-save
```

10.4 Reverse Path Filtering

The Linux kernel has a policy feature that can be set on a per-interface basis called the reverse path filter. When this feature is enabled, the source IP address for each packet received on enabled interfaces is inspected and looked up in the system routing table to validate that the source IP address is reachable through the same interface on which the packet was received. If it is not reachable through that interface, then the packet is discarded. This feature can be problematic for Dual Channel Wi-Fi because the data channel interfaces are not configured with an IP address. They have no routes or reachable addresses configured in the system routing table, so Linux will discard all incoming traffic on the data channel interfaces. The provided example DCSTAD script disables reverse path filtering for all data channels—this function must not be overlooked when implanting a new DCSTAD script. The reverse path filter is adjustable either through a “sysctl” or in the procfs tree at the following path: /proc/sys/net/ipv4/conf/**{INTERFACE_NAME_HERE}**/rp_filter.

10.5 Wireshark

When debugging a network problem, it is best to run Wireshark to obtain visibility of what is happening with the network traffic itself.

10.5.1 Dual Channel Wi-Fi Wireshark Dissectors

A Wireshark patch is available for dissecting and debugging Dual Channel Wi-Fi signaling traffic. Dissectors are included for both the CL3 and DCW protocols.

10.5.2 Building a Dual Channel Wi-Fi Protocol-Aware Wireshark

In addition to the packages installed in Section 7.3.1, building Wireshark requires some additional package dependencies.

```
$ sudo apt-get install cmake libgcrypt20-dev libglib2.0-dev flex bison libgtk-3-dev libpcap-dev
```

To build the Dual Channel Wi-Fi protocol-aware Wireshark, simply check-out from the code repository and run “make”.

```
$ git clone git@github.com:cablelabs/dcwwireshark.git
$ cd dcwwireshark/
$ make
```

10.5.3 Running the Dual Channel Wi-Fi Protocol-Aware Wireshark

The DCW custom Wireshark can be run by executing the following commands.

First, change to the directory where Wireshark was built.

```
$ cd dcwwireshark/
```

Then, invoke the Makefile's "run" target with super-user permission.

```
$ sudo make run
```

Note: Running Wireshark requires an X11 graphical environment.

11 KNOWN ISSUES AND LIMITATIONS

11.1 Network Manager Conflicts

Almost every graphical desktop Linux distribution comes with a network manager enabled by default. The provided example DCSTAD script assumes that there is no network manager controlling the data channel interfaces. If it is desirable to use a network manager to control the data channel interfaces, the software architecture is adaptable, allowing a custom DCSTAD script to be implemented in order to integrate with an existing network manager.

* * *