Wireless
Wi-Fi

Dual Channel Wi-Fi Arris XB3 RDK-B Integration and Operations Guide

WR-GL-DCW-ARRIS-XB3-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-ARRIS-XB3-V01-190513			
Document Title:	Dual Channel Wi-Fi Arris XB3 RDK-B Integration and Operations Guide			
Revision History:	D01 – Released 01/18/19 V01 – Released 5/13/19			
Date:	May 13, 2019			
Status:	~~Work in Progress~~	~~Draft~~	Released	~~Closed~~
Distribution Restrictions:	~~Author Only~~	~~CL/Member~~	~~CL/ Member/ Vendor~~	Public

Contents

Figures

1 SCOPE

1.1 Introduction and Overview

This document describes how to integrate and use the Dual Channel Wi-Fi (DCW) feature on the Arris XB3 device for RDK-B.

1.2 Purpose of Document

The purpose of this document is to explain the process for integrating the Dual Channel Wi-Fi (DCW) feature into the Arris version of XB3 cable modem for RDK-B. In addition to the integration steps, a complete user manual and troubleshooting guide are included.

2 REFERENCES

2.1 Informative References

None

3 TERMS AND DEFINITIONS

This document uses the following terms.

ARM	A type of CPU core used on the XB3.
Arris	Hardware vendor.
ATOM	A type of CPU core used on the XB3.
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 that is the core kernel for RDK.
OSX	Proprietary operating system developed and owned by Apple.
primary channel	The main Wi-Fi connection used in Dual Channel Wi-Fi for both DCW signaling and upstream and downstream traffic.
TR-181	Data-model standard for TR-069.
Wi-Fi	A technology enabling the wireless transmission and reception of LAN traffic.
XB3	A model of cable modem manufactured by several vendors. This document refers specifically to the Arris version of the XB3, which is model TG1682G.

4 ABBREVIATIONS AND ACRONYMS

This document uses the following abbreviations.

AP	access point
API	application programming interface

CCSP	Common Component Software Platform
CPU	central processing unit
DCW	Dual Channel Wi-Fi
DCWAPD	Dual Channel Wi-Fi Access Point Daemon
DHCP	dynamic host control protocol
DOCSIS	Data Over Cable Service Interface Specification
IP	Internet Protocol
LAN	local area network
MAC	media access control
OID	object identifier
RAM	random access memory
RDK	Reference Design Kit
RDK-B	Reference Design Kit for Broadband
RSSI	received signal strength indicator
SNMP	simple network management protocol
SSID	service set identifier
TCP	Transmission Control Protocol
UART	universal asynchronous receiver-transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VAP	virtual access point
VLAN	virtual local area network

5 ARCHITECTURE OVERVIEW

The software component architecture of the Dual Channel Wi-Fi access point (AP) software is integrated into the XB3 as shown in Figure 1. Components in orange are added for Dual Channel Wi-Fi functionality.

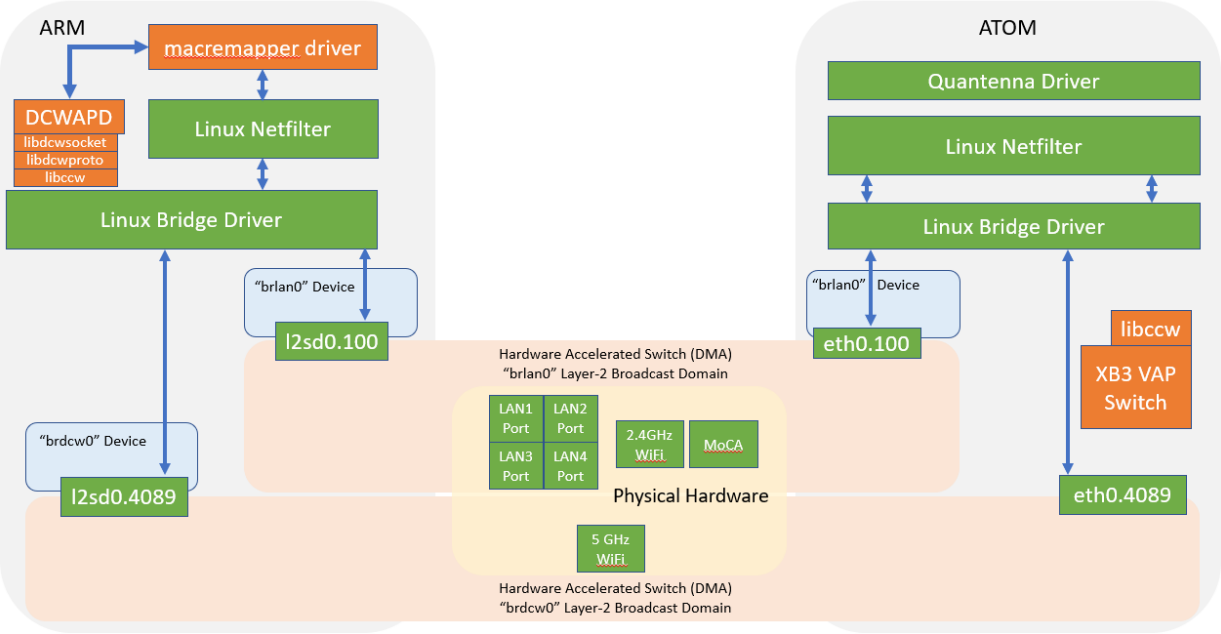

Figure 1 – Arris XB3 Dual Channel Wi-Fi Software Architecture

5.1 Overview of Dual Channel Software Component Bootup

5.1.1 ARM Core

The Dual Channel Wi-Fi bootup process on the ARM core is as follows:

- Initialize the data channel VLAN.
- Insert the MAC Address Remapper module into the running kernel.
- Start the DCWAPD service daemon.

5.1.2 ATOM Core

The Dual Channel Wi-Fi bootup process on the ATOM core is as follows:

- Initialize the data channel VLAN.
- Start the XB3 VAP Switch daemon.

5.1.3 Utopia Patches

The Utopia initialization script (/etc/utopia/utopia_init.sh) was patched for both target processors: ARM and ATOM. The files are appended to call an included "utopia_init_dcw.sh" script, which provides initialization procedures specific to Dual Channel Wi-Fi.

The primary purpose of the DCW initialization procedures in Utopia is to set up the data channel virtual local area network (VLAN). On the ARM core, the script also launches the DCWAPD daemon.

5.2 Data Channel Traffic Isolation (VLAN)

All data channel traffic is exclusively isolated and carried on a separate VLAN. The DCW data channel currently uses VLAN 4098; the number was chosen arbitrarily from VLANs not in use at the time the DCW AP code was developed. Any unused VLAN number can be used.

If desired, changing the VLAN number used for data channel traffic can be done by modifying the following files:

- utopia_init_dcw.sh—meta-cablelabs-dcw-xb3/recipes-dcw/utopiainitdcw/files
    - DATA_CHANNEL_VLAN_ID=4089
- main.cxx—xb3vapswitch/src
    - _dcwdataVlan = 4089;

5.3 Device Behavior Changes

Running an XB3 device image after integrating the Dual Channel Wi-Fi code has yielded no run-time behavior changes compared to running an image without this functionality. However, with the current hardware platform, the device's 5 GHz Wi-Fi radio will be unusable when the Dual Channel Wi-Fi feature is enabled with the default settings because the radio functions exclusively for the data channel.

Note: If desired, Dual Channel Wi-Fi can be configured to use the 2.4 GHz radio for the data channel instead of the 5 GHz radio. See Section 9.2.

6 HARDWARE COMPONENTS

This integration and operations guide applies only to the Arris cable modem model TG1682G because the Dual Channel Wi-Fi feature is tied to hardware features of the device such as the accelerated Ethernet switch. Refer to Section 8 for general guidelines on porting to other RDK-B platforms.

7 SOFTWARE COMPONENTS

7.1 Individual Components

Dual Channel Wi-Fi AP functionality comprises several individual 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 libccw

The *libccw* component is a C++ wrapper library for the TR-181/CCSP, which is called by the DCWAPD process and the XB3 VAP Switch components. The purpose of putting this functionality into its own library was to reduce the amount of redundant CCSP code in the business-logic components.

This component is used only on RDK-B with the AP code on both the ARM and ATOM cores.

Important note: The compiler flags used to build this component were obtained from an ARM libccsp_common compilation but seem to work on ATOM as well.

7.1.4 MAC Address Remapper

The *MAC Address Remapper* component provides functionality to the ARM-core Linux kernel, which is responsible for the actual filtering and switching of Ethernet traffic onto the data channel.

For more information on this component, please refer to the MAC Address Remapper documentation.

7.1.5 DCWAPD

The *DCWAPD* daemon component is the heart of the Dual Channel Wi-Fi business-logic implementation. The component runs on the ARM-core and is responsible for orchestrating all Dual Channel Wi-Fi operations. The component provides the "Device.DCW." parameters to the data model. All of the AP signaling logic is implemented in this component.

7.1.6 XB3 VAP Switch

The *XB3 VAP Switch* component is the component used to provide the DCWAPD process the means to isolate the data channel service set identifier (SSID) onto its respective VLAN. Although a better approach may be available based on recent versions of RDK-B, the RDK-B data model used does not provide parameters for the support. Use of this component should be reevaluated as the RDK-B platforms are updated.

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.

For the XB3 specifically, the RDK-B Yocto layer repository for Dual Channel Wi-Fi can be found at https://github.com/cablelabs/meta-cablelabs-dcw-xb3 (meta-cablelabs-dcw-xb3). This repository contains all of the Yocto recipes for cloning and building all software components necessary for building an XB3 image with Dual Channel Wi-Fi functionality.

7.3 Building

To compile the Dual Channel Wi-Fi functionality into the Arris XB3 image, follow the standard build procedure; deviations for Dual Channel Wi-Fi are shown in red, below.

For this example, we will use the example path "/build/arrisxb3" (substitute for your build path).

First, clone the initial repos.

```
$ mkdir /build/arrisxb3
$ cd /build/arrisxb3
$ repo init -u ssh://gerrit.teamccp.com:29418/rdk/yocto_oe/manifests/arris-intel-manifest -b
stable2 -m arrisxb3.xml --repo-url=ssh://gerrit.teamccp.com:29418/rdk/tools/git-repo --no-repo-
verify -g all
$ repo sync --no-tags
$ git clone git@github.com:cablelabs/meta-cablelabs-dcw-xb3.git
```

Then, build the ATOM image.

```
$ source meta-rdk/setup-environment build-xb3
  -- choose option 4 (arrisxb3atom.conf)
$ echo 'BBLAYERS =+ "${RDKROOT}/meta-cablelabs-dcw-xb3/bblayer-atom"' >> conf/bblayers.conf
$ bitbake comcast-broadband-dev-release-image
```

Finally, build the ARM image.

```
$ rm conf/bblayers.conf conf/local.conf conf/auto.conf
$ cd ..
$ source meta-rdk/setup-environment build-xb3
  -- choose option 1 (arrisxb3arm.conf)
$ echo 'BBLAYERS =+ "${RDKROOT}/meta-cablelabs-dcw-xb3/bblayer-arm"' >> conf/bblayers.conf
$ bitbake comcast-broadband-dev-release-image
```

7.4 Yocto Meta Layer Directory Structure

All build recipes and scripts are located in the "meta-cablelabs-dcw-xb3" repository. Recipes in this repository reference several software component repositories that are cloned during the bitbake build process. All software component repository git locations are referenced in a single file located in the repository root: "recipe-sources.inc". Having a single place to manage repo locations simplifies the process of migrating repositories in the future.

7.4.1 recipes-dcw/

Recipes under this directory pertain to building components specific to Dual Channel Wi-Fi.

7.4.1.1 dcwapd

Clones and builds the Dual Channel Wi-Fi access point daemon.

7.4.1.2 filters

Kits the sample traffic filter profile files into the image for use with DCWAPD.

7.4.1.3 libdcwproto

Clones and builds the DCW protocol C library. The DCWAPD component is statically linked with this recipe.

7.4.1.4 libdcwsocket

Clones and builds the DCW socket C library. The DCWAPD component is statically linked with this recipe.

7.4.1.5 snmp

Sample recipe demonstrating how to add a simple network management protocol (SNMP) parameter to enable/disable Dual Channel Wi-Fi. The included sample object identifier (OID) is for demonstration purposes only and is not intended for production use.

7.4.1.6 utopiainitdcw

Appends to the Utopia initialization scripts for both the ARM and ATOM cores. This recipe is responsible for bootstrapping the Dual Channel Wi-Fi functionality on the device.

7.4.1.7 xb3vapswitch

Clones and builds the XB3 VAP Switch helper daemon process.

7.4.2 recipes-kernel/

Recipes under this directory pertain to extending the Linux kernel functionality for Dual Channel Wi-Fi.

7.4.2.1 macremapper

This recipe includes two components: the “macremapper” Linux kernel component and the user-land debug utility “mrmctl”.

7.4.3 recipes-libccw/

Recipes under this directory pertain to building the CableLabs CCSP Wrapper library and its dependencies.

7.4.3.1 ffcall

Clones and builds libffcall, which is a dependency for compiling the CableLabs CCSP Wrapper C++ library.

7.4.3.2 libccw

Clones and builds the CableLabs CCSP Wrapper C++ library.

7.4.4 bblayer-arm/

Th *bblayer-arm* is the layer directory for providing the ARM core layers. Everything is symlinked except for the “recipes-layer/” directory. There is a package group append recipe within this path that is responsible for hooking the Dual Channel Wi-Fi recipes into the build.

7.4.5 bblayer-atom/

Th *bblayer-atom* is the layer directory for providing the ATOM core layers. Everything is symlinked except for the “recipes-layer/” directory. There is a package group append recipe within this path that is responsible for hooking the Dual Channel Wi-Fi recipes into the build.

8 PORTING TO OTHER PLATFORMS

Because the Dual Channel Wi-Fi feature has dependencies on the hardware platform, it is not possible to provide a code-base that will universally compile and run across all RDK-B devices. However, most of the software components are reusable across all platforms.

8.1 Software Component Portability/Reusability

The following software components are virtually portable to any RDK-B platform and should be reusable with little to no modifications.

- libdcwproto
- libdcwsocket
- libccw
- MAC Address Remapper

The following software component is mostly reusable but will require adaptation for the target platform.

- DCWAPD

The following software components are highly likely to not be reusable for a new platform and will require a rewrite or another solution architecture.

- XB3 VAP Switch
- platform script (e.g., Utopia patches)

8.2 Considerations for Porting to Another RDK-B Platform

1. Determine how to intercept and isolate data channel traffic within its own layer-2 broadcast domain. The MAC Address Remapper component can be used for this purpose when layer-2 traffic can be passed through a Linux bridge device.
2. A version of the DCWAPD software component, which implements the platform-specific operations, is required for the target platform. Most, if not all, of the core DCWAPD code, including the business-logic, should be reusable. The TR-181 data-model provider code can be referenced and copied from the XB3 implementation.
3. Implement platform-specific bring-up scripts and build recipes.

9 USER GUIDE

9.1 Data Model

All Dual Channel Wi-Fi configuration parameters and status can be found in the TR-181 data model under the "Device.DCW." subtree.

9.1.1 Configurable Parameters

Located under **Device.DCW.**:

- **FilterPath** – File-system path indicating where the DCW traffic filter profile files can be found.
- **Network.1.** – Configuration parameters pertaining to a single DCW network.
    - **Enable** – Enables/disables DCW functionality on this network.
    - **PrimaryChannelAPIndex** – The access point to use for the primary channel (see Section 9.1.3).
    - **DataChannel.1.APIndex** – The access point to use for the data channel (see Section 9.1.3).
- **StationFilterOverride.X.** – Configuration parameters for a single station/filter association ("X" is the index).
    - **PrimaryMACAddress** – The primary channel MAC address to use for applying the configured filter name.
    - **FilterName** – The name of the traffic filter profile (see "Device.DCW.Filter.") to apply.

9.1.2 Status/Telemetry Parameters

Located under **Device.DCW.**:

- **Filter.X.Name** – Name of a single traffic filter profile detected in the filter path.
- **Station.X.** – DCW state for a single station ("X" is the index).
    - **PrimaryMACAddress** – The primary MAC address of the station (used for signaling).
    - **NetworkIndex** – The network index in "Device.DCW.Network." that this station is attached to (always 1 for XB3).
    - **FilterName** – The name of the filter currently applied for this station (can be empty during bonding).
    - **BondedDataChannel.X.** – A single data channel bond for this station; may be many ("X" is the index).
        - **APIndex** – The access point that this station's data channel is associated with.
        - **MACAddress** – The MAC address of this station's data channel.

9.1.3 Parameter Value Format

MAC address values may be delimited with either "-" or ":" characters. Uppercase and lowercase characters are also acceptable. For example, both "00-11-22-33-aB-Cd" and "00:11:22:33:Ab:cD" are acceptable.

AP index values reference the "Device.WiFi.AccessPoint.X" nodes in the data-model; "X" is the index value.

9.2 Common Task Examples

This section includes a few examples that demonstrate how to perform common Dual Channel Wi-Fi tasks by using the TR-181/CCSP data-model commands. These commands may be executed from either the ARM or ATOM universal asynchronous receiver-transmitter (UART) console on the device.

9.2.1 Enable and Disable Dual Channel Wi-Fi

WARNING: Ensure that different SSIDs are configured for the 2.4 GHz and 5 GHz radio before enabling.

Execute this command to enable Dual Channel Wi-Fi.

```
$ dmcli eRT setv Device.DCW.Network.1.Enable bool true
```

To disable, execute the same command, changing "true" to "false".

```
$ dmcli eRT setv Device.DCW.Network.1.Enable bool false
```

9.2.2 List Active Dual Channel Stations

```
$ dmcli eRT getv Device.DCW.Station.
```

9.2.3 List Detected Traffic Filter Profile Names

```
$ dmcli eRT getv Device.DCW.Filter.
```

9.2.4 Force a Station to Use a Specified Traffic Filter Profile

For this example, the station with a primary channel MAC address of "00-11-22-33-44-55" is forced to always be assigned a traffic filter profile "All_TCP". This example assumes "Device.DCW.StationFilterOverride.1." is the index added. Please note the output after executing the "addtable" command.

```
$ dmcli eRT addtable Device.DCW.StationFilterOverride.
$ dmcli eRT setv Device.DCW.StationFilterOverride.1.PrimaryMACAddress string 00-11-22-33-44-55
$ dmcli eRT setv Device.DCW.StationFilterOverride.1.FilterName string All_TCP
```

9.2.5 Configure the 2.4 GHz Radio as the Data Channel

The default setup is to use "Device.WiFi.AccessPoint.1." for the customer 2.4 GHz AP and "Device.WiFi.AccessPoint.2." for the customer 5 GHz AP. Therefore, to change the Dual Channel Wi-Fi configuration to use the 2.4 GHz radio as the data channel, simply set the data channel AP index to 1.

```
$ dmcli eRT setv Device.DCW.Network.1.DataChannel.1.APIndex uint 1
```

9.2.6 Configure the 5 GHz Radio as the Data Channel

The out-of-box default configuration for the data channel is to use 5 GHz radio. However, if this was changed, simply reset the data channel AP index to 2.

```
$ dmcli eRT setv Device.DCW.Network.1.DataChannel.1.APIndex uint 2
```

9.3 Traffic Filter Profiles

Currently, all traffic filter profiles are delivered with the software image. The default path for the files is "/var/run/dcw/filters/" (Figure 2).

```
root@Docsis-Gateway:/# dmcli eRT getv Device.DCW.FilterPath
CR component name is: eRT.com.cisco.spvtg.ccsp.CR
subsystem_prefix eRT.
getv from/to component(eRT.com.cisco.spvtg.ccsp.dcw): Device.DCW.FilterPath
Execution succeed.
Parameter    1 name: Device.DCW.FilterPath
               type:     string,    value: /var/run/dcw/filters

root@Docsis-Gateway:/# ls -la /var/run/dcw/filters
drwxr-xr-x    2 root     root           100 Jan  1  1970 .
drwxr-xr-x    3 root     root           120 Jan  1  1970 ..
-rw-r--r--    1 root     root            75 Jan  1  1970 All_TCP.tfp
-rw-r--r--    1 root     root           118 Jan  1  1970 In_HTTP_HTTPS.tfp
-rw-r--r--    1 root     root            81 Jan  1  1970 TFP_Default.tfp
root@Docsis-Gateway:/#
```

Figure 2 – ARM UART Output Depicting Filter Path and Files

Because this is a tmpfs (RAM disk) location, filters can be modified, added, or removed during system operation. There is no requirement, however, for the traffic filter profile files to reside in a writable location, and they may be kitted in a read-only part of the system image if desired.

9.3.1 List Detected Filters

The AP daemon process scans the filter directory for filter files matching the "*.tfp" (traffic filter profile) file extension. For each matching filename, the daemon opens and validates the filter file. If all is good, the filter is listed in the data model.

Execute this command to display the validated filters that the AP daemon is aware of.

```
$ dmcli eRT getv Device.DCW.Filter.
```

When the AP is running, the filter will also appear in the running MAC address remapper configuration, which can be displayed by issuing the following command.

```
$ mrmctl show
```

9.3.2 Add a Sample Filter Live

The following example supposes is a cloud-based service that delivers traffic from TCP port 1234. This service is available on the entire IP block: 10.20.30.40/29. To prioritize this service for all Dual Channel Wi-Fi stations, move all downstream traffic to the data channel.

The filter would be created as follows.

```
$ echo '*:10.20.30.40/29:1234:tcp:*' > /var/run/dcw/filters/MyService.tfp
```

Note: If Dual Channel Wi-Fi is already running, it may need to be restarted.

```
$ dmcli eRT setv Device.DCW.Network.1.Enable bool true
```

At this point, the filter is ready for use. See Section 9.2.4 for an example showing how to associate the filter with specified stations.

10 TROUBLESHOOTING

In general, when troubleshooting any sort of network issue, Wireshark is the recommended tool to use as it provides visibility to what traffic is passing through the network. For information on the protocol format of the signaling messages, it is advised to review the Dual Channel Wi-Fi protocol specification document.

10.1 Useful Wireshark Filters

- Show only Dual Channel Wi-Fi signaling messages on a Wi-Fi adapter not in monitor mode:
  - `eth.type == 0xb4e3`
- Show only Dual Channel Wi-Fi signaling messages on a Wi-Fi adapter in monitor mode:
  - `llc.type == 0xb4e3`

10.2 Validate the Health of All Software Components

To confirm the overall health of the Dual Channel Wi-Fi components on the device, the following basic steps should be executed, yielding no errors.

10.2.1 Ensure the DCWAPD Process Is Functional

To determine if the Dual Channel Wi-Fi AP daemon process is running, query the DCW parameters in the data model.

```
$ dmcli eRT getv Device.DCW.
```

10.2.2 Ensure the XB3 VAP Switch Process Is Functional

Querying the DCWXVS parameters in the data model will indicate if the XB3 VAP Switch daemon process (workaround) is running.

```
$ dmcli eRT getv Device.DCWXVS.
```

10.2.3 Ensure the MAC Address Remapper Is Functional

Query the state of the MAC Address Remapper using the “mrmctl” utility:

```
$ mrmctl show
```

If it is not desired to kit the “mrmctl” utility into the image, the state of the MAC Address Remapper can be queried by calling “cat” on the control device.

```
$ cat /proc/macremapctl
```

10.3 Debugging the MAC Address Remapper

Because the MAC Address Remapper is the component responsible for the actual traffic flow of Dual Channel Wi-Fi, it is imperative to have the means to troubleshoot it. The “mrmctl” utility is a command-line tool that allows the developer to debug the state of the MAC Address Remapper. Everything exposed through the software API is also exposed through the “mrmctl” utility. For more information on this process, please refer to the MAC Address Remapper documentation.

10.4 Station Does Not Bond in Dual Channel Wi-Fi Mode

- Confirm that the station is correctly joined to the primary channel SSID. If more than one AP is using the primary channel SSID, then it is highly advised to change the primary channel SSID to something unique to only this AP.
- Ensure the Dual Channel Wi-Fi client software is running and that there are no errors.
- Ensure the station's Wi-Fi adapters support the radio frequencies and Wi-Fi standards being used. For example, using a USB Wi-Fi dongle that supports only 2.4 GHz for a data channel on 5 GHz will not work.
- Run Wireshark on the station's primary channel interface to validate the presence of the DCW signaling frames.
- Check the data model on the AP for presence of the station's primary channel MAC address.
- Try swapping the station's Wi-Fi adapters. For example, if the internal adapter is used as the primary channel, try using the USB dongle as the primary channel adapter.

10.5 Station Bonds in Dual Channel Wi-Fi Mode but Traffic Does Not Flow

- Ensure the primary channel SSID and data channel SSID are different.
- Completely disable Dual Channel Wi-Fi on the AP, and test both radios individually.
- Check the station's RSSI values. If too low, try reducing the channel width of the data channel.

11 KNOWN ISSUES AND LIMITATIONS

11.1 Configuration Persistence

Currently, Dual Channel Wi-Fi functionality is not persistent across reboots of the XB3; that is, the Dual Channel Wi-Fi functionality must be enabled in the data model per each reboot. This feature allows the operator to control when Dual Channel functionality is enabled. This functionality can be enabled through the DOCSIS configuration file once the SNMP OIDs are determined.

11.2 SSID for Primary Channel and Data Channel Must Be Different

Because of the way the Dual Channel Wi-Fi signaling protocol works, it is imperative that the primary channel and data channel SSIDs are different. If these values are the same, the stations will unlikely be able to successfully complete channel bonding.

11.3 Filtering DHCP Traffic Through the Data Channel

Most dynamic host control protocol (DHCP) client implementations bind to a specific interface when operating. Filtering DHCP traffic (UDP ports 67 and 68) through the data channel can cause the client to not receive DHCP packets. If this happens, Dual Channel Wi-Fi stations may not be able to obtain or renew existing DHCP leases, therefore cutting off IP communication with the network. For this reason, it is not advisable to filter DHCP traffic onto the data channel.

11.4 LAN-to-LAN Traffic

Because of the function of the hardware switch on the XB3, not all traffic can be filtered and put onto the data channel. Traffic originating on the same layer-2 broadcast domain shared with the Wi-Fi AP often does not pass through the CPU, so the Dual Channel Wi-Fi software cannot intercept the traffic for filtering.

The following example (Figure 3) supposes a scenario in which there are two devices on the local area network (LAN): a desktop PC connected through Ethernet and a laptop PC connected through Dual Channel Wi-Fi.

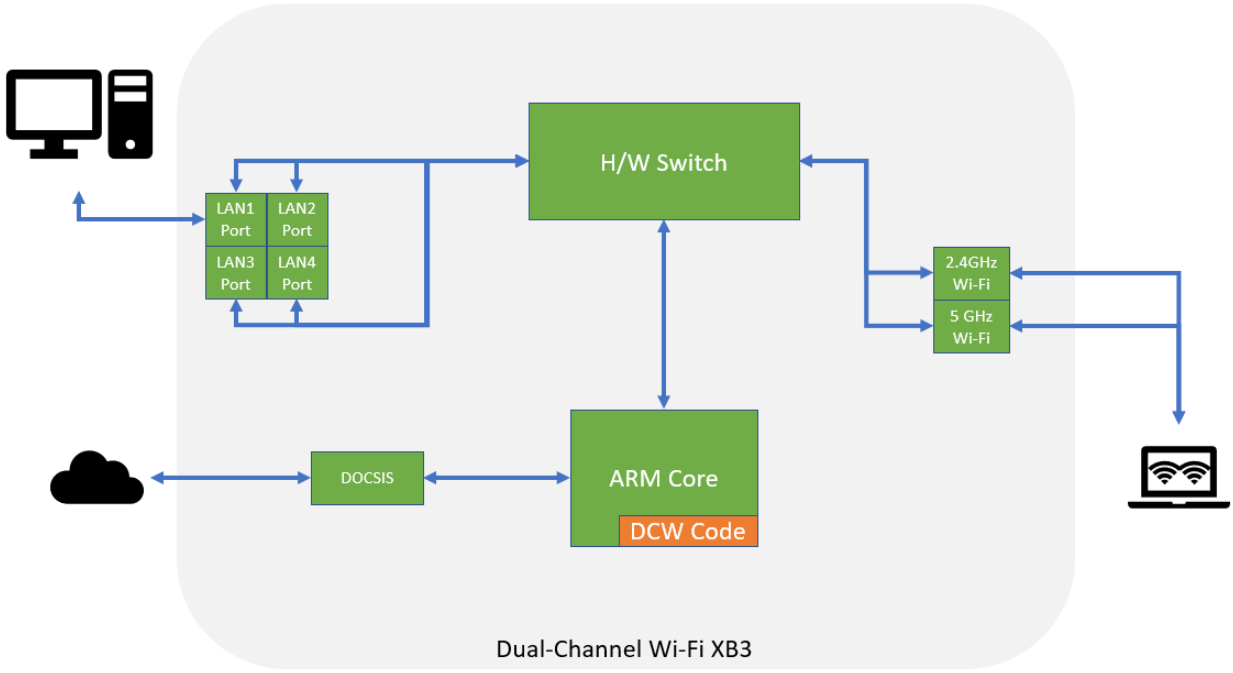

Figure 3 – Example of LAN-to-LAN Traffic Flow

The desktop PC and laptop PC can communicate normally with each other because they are on the same layer-2 broadcast domain. However, traffic sent from the desktop PC to the laptop PC will only be transmitted through the primary channel because the traffic never gets to the Dual Channel Wi-Fi code running in the ARM core. All internet traffic going through the DOCSIS port does go through the ARM core and therefore will be subject to Dual Channel Wi-Fi traffic filtering.

* * *