Troubleshoot Wireless Client Interoperability Issues with CUWN

Contents

Introduction

This document describes interoperability issues when they arise with Cisco Unified Wireless Network (CUWN) solution.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Cisco Wireless APs
- Wireless LAN Controllers (WLC)
- Related Network Devices

Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

> **Note**: The intended audience for this document are experienced wireless network engineers and administrators who are already familiar with the use, configuration and troubleshooting of these topics.

Background Information

It can be common to find that given the various client devices that both exist and continue to be developed. A variety of issues can arise with regards to establish, maintain, or simply to get the most out of their connection to the wireless network and to support infrastructure.

This can often come down to a simple configuration issue on the part of the client device and/or the wireless infrastructure itself. However, in some cases this can be attributed to an interoperability issue with regards to a specific client device and components that support it (supplicant, WLAN adapter, wireless driver,...), and/or the APs in question. As wireless engineers, such interoperability issues pose an opportunity to identify, troubleshoot, and resolve potentially complex challenges.

This document describes in detail what information needs to be initially collected to effectively investigate and troubleshoot such wireless interoperability issues when they arise with Cisco's Unified Wireless Network (CUWN) solution. The need for such a comprehensive approach becomes increasingly important with the ever growth in numbers and combinations of wireless client devices and access point (AP) radios.Additional information to what is outlined in this article might be requested and needed to be collected on a case by case basis, given the unlimited number of variables that might dictate such requirements. However, the information detailed here is a generic guideline to address any potential wireless client interoperability issue.

I. Problem Definition

The first step to effectively approach any problem with the intent to get resolute, is to accurately define the issue at hand. To do so, ensure that at a minimum of these questions are asked and their answers are clearly documented:

- Is the issue restricted to a specific model of APs and/or radio type (2.4 GHz versus 5 GHz)?
- Is the issue observed only on specific version(s) of WLC software?
- Is the issue experienced with only specific version(s) of client type(s) and/or software (OS version, WLAN driver version,...)
- Are there any other wireless devices which do not experience this issue? If so, what are they?
- Is the issue reproducible while the client is connected to a simplified wireless setup such as an open SSID, with a channel width of 20 MHz, and 802.11ac disabled? (that is, does the issue happen on 802.11n mode versus 802.11ac mode only?).
- If the issue is not reproducible with an open SSID, at what minimum security configuration is the issue seen? (PSK or 802.1X on the WLAN).
- What were the previous known-good configuration and software versions?

II. WLC Configuration And General Logs

Run-Config

Without exception, it is of absolute necessity to collect the WLC configuration for a detailed review of features used by the customer, their specific setup, and other such details. To do so, you must establish a Telnet/SSH session to the WLC(s) in question and save the output of these CLI commands to a text file:

```
config paging disable

show run-config
```

The full run-config output is always preferred, as it includes detailed information with regards to the joined APs and associated RF information. Though in some cases and situations, such as when you initially work with a WLC with a large number of APs joined (8510 WLC with 2500+ APs). It might be preferred to initially collect just the configuration of the WLC without such AP information for quick review, as the full show run-config might take 30 minutes or more to complete the given the number of APs. However, it might still be needed to collect the full run-config output at a later time.

To do so, you can optionally collect the output of these CLI commands to a text file:

```
config paging disable

show run-config no-ap

show wlan apgroups
```

WLC Configuration File

In addition to either the **show run-config** or **show run-config no-ap** output, it is also recommended to collect a full backup of the WLC configuration as well. This is of assistance, if a lab recreate needs to be conducted by both TAC/HTTS and BU Escalation, to try and reproduce the issue in a Cisco lab environment. A backup of the WLC can be collected via the GUI or the CLI of the WLC in question, with the use of either TFTP or FTP to save the configuration file to the external TFTP/FTP server. This example shows the usage of both the GUI and CLI to save a backup of the WLC, with the use of TFTP:

GUI

Commands > Upload File > Configuration > Upload as shown in the image.

CLI

```
transfer upload datatype config

transfer upload mode tftp transfer upload serverip <TFTP-Server_IP-address> transfer upload path
/ transfer upload filename <desired-filename> transfer upload start
```

Syslogs from the WLC

At this time, you also want to collect the current logs from the WLC for additional review as needed. Ideally, you want to collect these logs immediately after your test with a wireless client whereby the reported issue is reproduced. If the customer exports the WLC logs to an external syslog server, then you want to retrieve them from there. Otherwise, you can save the msglog and traplog currently stored locally on the WLC by saving this CLI session output to another text file:

```
config paging disable

show msglog

show traplog
```

III. Client Device Details and Information

The next step is to gather as much information and specifics with regards to the client device(s) in use that experience a potential wireless interoperability issue. Such information must include, but is not necessarily limited to these:

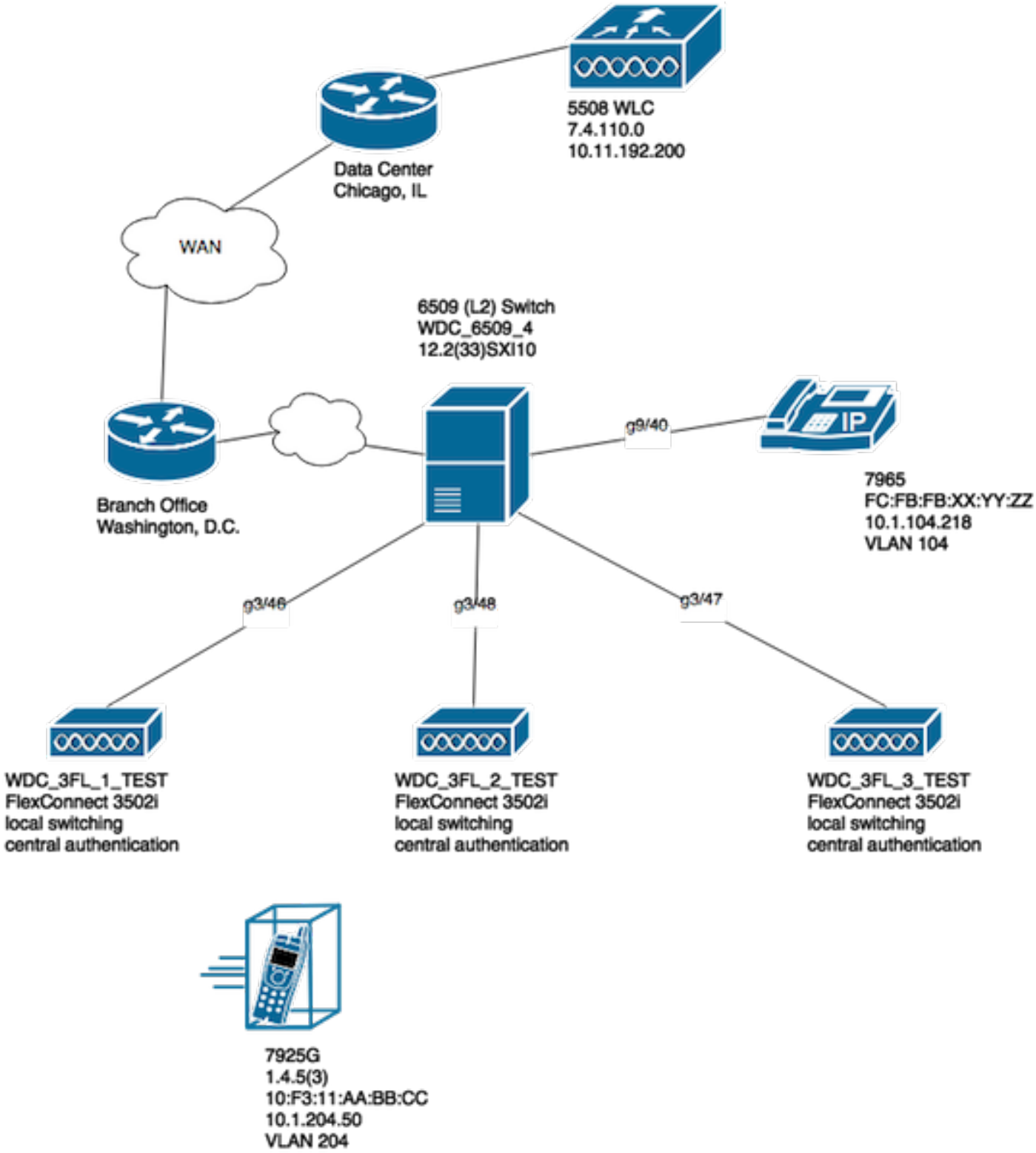
- Client type (tablet, smartphone, notebook PC,...)
- Device make and model
- OS version
- WLAN adapter model
- WLAN adapter driver version
- Supplicant used (Windows Zero Config / Auto Config, Intel PROSet,...)
- Security configured for use by the wireless client and WLAN (Open, PSK, EAP-PEAP/MSCHAPv2,...)
- Note any client parameters that have been changed from the default settings provided by the vendor in question (sleep state, roaming parameters, U-APSD,...).

Note: Any additional information or notes with regards to the client device(s) up to which includes screenshots of its WLAN related configuration(s), and so forth must also be included as needed.

IV. Network Topology

To further expedite troubleshooting efforts and the Root Cause Analysis (RCA) process, it is always recommended to provide a detailed and thorough network topology diagram. The network topology diagram must not only include details about the network and wireless infrastructure, but also provide an insight into the wireless device(s) in question which operates within the network (printers/scanners, what client VLAN(s) are in use,...) and their location(s) relative to one another.

A number of tools (Microsoft Visio, draw.io,...) and a variety of styles can be used to create such a network diagram. The important aspect is to simply ensure that the proper information is clearly reflected in the diagram provided for review by all involved parties and vendors. An example network topology that captures basic, but useful information with regards to both the infrastructure and client devices as shown in the image.



V. Track Additional Details and the Specifics

To help ensure that the appropriate information is collected at the time of any test with the client device(s) that end users experience issues with. It is recommended to preemptively create a spreadsheet or similar to record all the client issues and related details observed at the time of the test, such as this example:

MAC Address	Username	Description of Reported Symptom	Time end user observed symptom	Ping default gateway Y/N	WiFi Signal status (Connected/trying to Connect)	Record ipconfig /all (or e

xxyy.aabb.0011 test_user1	Intermittently disconnects from access point.	Lost network connectivity and wireless association from AP3.	N	Trying to Connect	ifconfig en0 en0: flags=8863<UP,BROAD mtu 1500 ether xx:yy:aa:bb:00:1 inet6 fe80::848:cb8f:88 inet 192.168.10.237 ne nd6 options=201<PER media: autoselect status: active

The goal of this exercise is to help document and determine a common pattern of interest, as well as to get an accurate picture of the issue(s) at hand. Once this spreadsheet is prepared to be used for data collection, you are now ready to begin your tests. Some additional, yet important considerations are as follows:

> **Note**: All debugs and packet captures collected need to be synchronized to the same NTP server for easier correlation with the logs, and must be taken at the same time for any given test.

> **Note**: Provide an accurate timestamp of when the issue is observed, and when the issue seems to recover (if applicable).

> **Note**: Always collect debugs filtered per client MAC address on both the AP and WLC.

> **Note**: Do not run show and debug commands on the AP within the same Telnet/SSH/console session, these are done separately in a different session accordingly.

> **Note**: AP debugs are preferred to be taken on Telnet/SSH versus Console, as the console is typically too slow to be effective.

VI. WLC - Show and Debug Commands

When tests are conducted to reproduce and troubleshoot potential wireless client interoperability issues, it is imperative that debugs and additional logs be collected from the wireless infrastructure in use. These two sections can explain in detail the specific logs and initial debug output that is collected from the WLC and AP(s), respectively.

WLC Debug Commands

```
config sessions timeout 0
debug client <MAC_address> debug dhcp message enable
```

With respect to the nature of the issue at hand, you can also add these WLC debugs on a case by case basis:

- **debug aaa detail enable** - use this if there are authentication related issues with the AAA

server
- **debug aaa events enable** - use this if there are authentication related issues with the AAA server
- **debug aaa all enable** - use this for auth issues; the output for this debug is verbose so use it only when absolutely needed (for AAA override cases,...)
- **debug mobility handoff** - use when there are roaming issues between WLCs

Once the issue is reproduced with the wireless client in question, and all of the information outlined in the sections prior and after this are collected and documented. In order to execute these CLI commands, you must disable the debugs on the WLC.

```
debug disable-all
```

WLC Show Commands

```
config paging disable

show time

show client detail <MAC_address>

ping <client_IP-address> <repeat count [1-100]>
```

As previously mentioned, ensure to run the WLC debugs in one Telnet/SSH session and collect the output for these show commands in another Telnet/SSH to the WLC. You must do the same to collect the AP debugs and show commands output detailed in these section.

VII. AP - Show and Debug Commands

Lightweight Cisco IOS® Access Points

Before you start any debugs on any lightweight Cisco IOS® AP(s) involved in the test, such as the 2600, 2700, 3700 or prior model Cisco access points. You must first execute these CLI commands on the AP, in order to avoid a timeout at the time of a Telnet/SSH/console session to the AP(s) in question when your client test(s):

```
debug capwap console cli

config t

line vty 0 4

exec-timeout 0

session-timeout 0
```

You can also follow these steps to use the console connection and replace the **line vty 0 4** statement with **line console 0** instead, in order to disable the exec and session timeouts for a serial/console connection accordingly.

- line console 0 - use to modify serial session timeout parameters
- line vty 0 4 - use to modify Telnet/SSH session timeout parameters

AP Show Commands

Before you begin the test, you must first collect a sample of these show commands on the AP. Collect the output of these show commands at least twice for each test which involves the wireless client in question; both before and after the test is complete.

```
term len 0

show clock

show tech

show capwap client mn

show int do1 dfs

show logging

more event.log

show trace dot11_rst display time format local

show trace dot11_rst

show trace dot11_bcn display time format local

show trace dot11_bcn
```

AP Debug Commands

Once you have collected the initial output of the aforementioned show commands, you can now enable the debugs on the same access point in a separate Telnet/SSH session as shown. Ensure to save the entire output to a text file.

```
debug dot11 {d0|d1} monitor addr <client_MAC-address>

debug dot11 {d0|d1} trace print clients mgmt keys rxev txev rcv xmt txfail ba

term mon
```

Flag	Description
d0	2.4 GHz radio (slot 0)
d1	5 GHz radio (slot 1)
mgmt	Trace management packets
ba	Trace Block ACK information
rcv	Trace received packets
keys	Trace set keys
rxev	Trace received events
txev	Trace transmit events
txrad	Trace transmit to radio
xmt	Trace transmit packets
txfail	Trace transmit failures
rates	Trace rate changes

To disable the debugs on the AP once the test and data collection process is completed, you can execute this CLI command on the AP:

```
u all
```

AP-COS Access Points

For 802.11ac wave 2 capable access points and later, such as the 1800, 2800 and 3800 model access points. These newer model APs introduce a completely new operating system for the access point platforms referred to as AP-COS. As such, not all commands as previously used on the traditional lightweight Cisco IOS® based access points as previously detailed still apply. If when you troubleshoot an issue involves interoperability issue with various client STA devices and AP-COS model APs, then these information must be collected from the AP-COS access point(s) involved with the equivalent test.

Before you start any debugs on any AP-COS model AP(s) involved in the test. You must first execute these CLI command on the AP, in order to avoid a timeout at the time of a Telnet/SSH/console session to the AP(s) in question when your client test(s):

```
exec-timeout 0
```

AP-COS Show Commands

Before you begin the test, you must first collect a sample of these show commands on the AP. Collect the output of these show commands at least twice for each test which involves the wireless client in question; both before and after the test is complete.

```
term len 0

show clock show tech

show client statistics <client_MAC-address>

show cont nss status

show cont nss stats

show log
```

1800 series | AP-COS Debug Commands

These debugs are specific to the 18xx series of access points. This is due to the fact that the chipset(s) used for the 1800 series of APs differ from those found in the 2800/3800 series access points, and thus a different set of debugs are required in this scenario by comparison. The corresponding debugs for the 2800/3800 series APs is covered in the next section.

Once you have collected the initial output of the aforementioned show commands, you must now enable the debugs on the same 1800 access point(s) in a separate Telnet/SSH session as shown. Ensure to save the entire output to a text file.

```
debug dot11 client level events addr <client_MAC-address>

debug dot11 client level errors addr <client_MAC-address>

debug dot11 client level critical addr <client_MAC-address>
```

```
debug dot11 client level info addr <client_MAC-address>

debug dot11 client datapath eapol addr <client_MAC-address>

debug dot11 client datapath dhcp addr <client_MAC-address>

debug dot11 client datapath arp addr <client_MAC-address>
```

In some cases, you might need to also enable the additional debugs on the 18xx AP to further troubleshoot client interoperability issues. However, this should be done only if/as requested by a Cisco TAC engineer for a corresponding service request/case.

As additional debugs might not only be far more verbose in their output but can also introduce additional load on the AP as well hence it requires additional time for proper analysis. Which under certain conditions can potentially disrupt service, if many client devices attempts to connect to the same AP under test or similar variables.

To disable the debugs on the AP-COS variant access point - whether on an 1800 or 2800/3800 series AP - once the test and data collection process is completed, you can execute this CLI command on the AP:

```
config ap client-trace stop
```

2800/3800 Series | AP-COS Debug Commands

Once you have collected the initial output of the aforementioned show commands, you must now enable the debugs on the same 2800/3800 access point(s) in a separate Telnet/SSH session as shown. Ensure to save the entire output to a text file.

```
config ap client-trace address add <client_MAC-address>
config ap client-trace filter all enable
config ap client-trace output console-log enable
config ap client-trace start
term mon
```

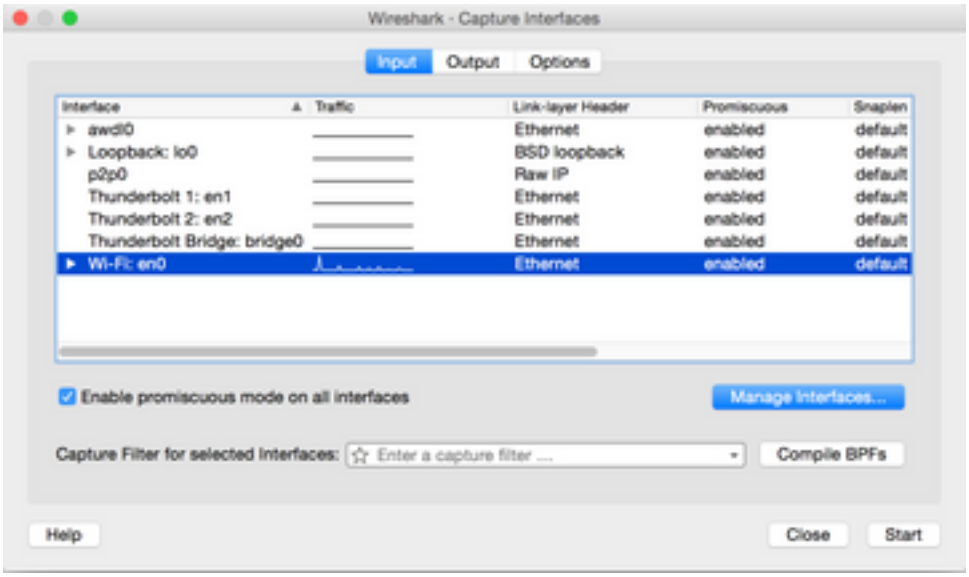
To disable the debugs on the 1800/2800/3800 series AP once the test and data collection process is completed, you can execute this CLI command on the AP:

```
config ap client-trace stop
```

VIII. Client Side Packet Captures

From the client device in use if it is a notebook PC, MacBook or similar, you must collect the promiscuous mode packet capture from the wireless interface of the client device used to reproduce the issue. Common utilities like Netmon 3.4 (Windows only) or Wireshark can be readily downloaded and used to collect this capture and save it to a *.pcap file. It depends on the device, there might also be means to collect a tcpdump or similar from the client in question, so you might need to consult with the client device manufacturer for assistance in this regard.

Here is an example to configure a Wireshark capture for the wireless interface on a MacBook Pro:



As with any packet capture, regardless of what utility is used to collect it, ensure to save the file in a pcap file format (*.pcap, *.pcapng, *.pkt,...). This is to ensure that not only Cisco engineers in any department can view the packet capture files with ease, but engineers from other vendors and organizations as well (Intel, Apple,...). This allows for a more seamless cooperation and collaboration process,which further facilitates both Cisco and the client device vendor(s) to better work together to investigate and resolve any potential interoperability issues.

IX. Over-the-Air (OTA) Packet Captures

In order to effectively troubleshoot any potential or existing wireless interoperability issues, it is crucial to collect a quality OTA packet capture of the issue. This allows for the detailed analysis of the actual 802.11 wireless communication between the wireless client and access point radio(s) in question, in addition to give further perspective to the client side and wireless infrastructure logs, and debugs. This is a critical step that must be accomplished for every test of a potential wireless interoperability issue, without exception.

However, often times the end customer is not properly equipped or prepared to collect OTA packet captures. This is a common obstacle that wireless engineers often face, and they must work with the customer to overcome this in a variety of ways. This article from the Cisco support forums can serve as a good start point to help guide and educate the customer accordingly:

802.11 wireless sniffing / packet capture

It is of paramount importance that the OTA packet capture(s) be collected in a pcap file format (*.pcap, *.pcapng, *.pkt,..), and includes 802.11 meta data (RSSI, channel, data rate,..). The OTA sniffer must also be kept in close proximity to the client device in question at all times during the test(s), to ensure an accurate perspective of the traffic sent and received to/from the client device tested.

> **Note**: If the test(s) in question involve a client device roaming scenario, whereby more than one 802.11 channel needs to be monitored in an aggregated packet capture. Then it is not currently recommended to use AirMagnet WiFi Analyzer from Fluke Networks.

The reason for this is due to the fact that aggregated packet captures with the use of this utility are currently saved in a proprietary file format, and not in a pcap style format that can be readily

viewed in Wireshark or other similar utilities. Ensure that your OTA packet capture is in a non-proprietary file format, this helps ensure that all parties and vendors involved can readily review any capture files at all times, and ultimately help expedite any resolution efforts.

in a format that is readable by current Wireshark, and that includes 802.11 meta data (RSSI, channel, data rate) - See more at: https://supportforums.cisco.com/document/75331/80211-wireless-sniffing-packet-capture#sthash.Xhlx5LSS.dpuf
Here are some common methods to collect an OTA packet capture:

- AirPCAP with Wireshark
- MacBook Pro
- OmniPeek Professional, OmniPeek Enterprise,...
- OmniPeek Remote Assistant (ORA)
- Cisco AP in Sniffer mode

802.11n Captures

For OTA packet captures which involves 802.11n wireless clients, there is at present more flexibility and ease of use. This is due to a wider variety of available wireless USB WLAN adapters that can be readily used with a number of tools, such as OmniPeek and others.

Do take note as to how the capabilities of the specific wireless adapter(s) used to collect an 802.11n OTA capture compare with the capabilities of the actual WLAN chipset used by the client device(s) which you attempt to troubleshoot. For instance, if the client device experiences a potential wireless interoperability issue which uses a 2 spatial stream (2SS) capable 802.11n chipset. Then it is highly recommended to ensure that the wireless adapter used to collect an OTA packet capture is also a 2SS or better adapter, with 802.11n or newer specifications.

802.11ac OTA Captures

For 3 spatial stream (3SS) 802.11ac captures, you can use the native sniffing capabilities of a 2014 model MacBook Pro or later with Mac OS X 10.10.x or higher. If troubleshooting a 2 spatial stream 802.11ac client device, you can also use a MacBook Air for 802.11ac captures. The Air model of MacBooks use 2SS only WLAN chipsets currently at the time of this writing. You can refer to the listed Cisco support forums article for instructions on how to collect OTA packet captures with the use of Mac OS X, through a variety of methods:

Wireless Sniffing with the use of Mac OS X 10.6+

You can also use either a 2702/2802/3702/3802 series or similar AP in sniffer mode to collect a proper 802.11ac packet capture with 3SS. You can also refer to the listed resource for a current list of available 802.11ac wireless adapters. Some of which are able to potentially be used with common tools like OmniPeek and others to collect an 802.11ac packet capture (chipsets from Ralink, Atheros,...):

https://wikidevi.com/wiki/List_of_802.11ac_Hardware#Wireless_adapters

You can also use either a 2702/2802/3702/3802 series or similar AP in sniffer mode to collect a proper 802.11ac packet capture with 3SS. For convenience, step by step instructions on how to configure a Cisco AP in sniffer mode and collect an OTA packet capture can be found in the Cisco support forums article:

Cisco AP in Sniffer mode

For troubleshooting roaming scenarios with a wireless client device, the common challenge is to effectively collect an OTA packet capture across multiple channels. This method of simultaneously monitoring multiple 802.11 channels is achieved by the collection of aggregated OTA packet capture. It is recommended to use multiple, compatible 802.11ac capable USB WLAN adapters with a compatible network analysis software in order to achieve this. Some common 802.11ac capable USB WLAN adapters include the Savvius WiFI Adapter for OmniPeek (802.11ac), Netgear A6210, or similar.

X. Summary

Here is a brief recap of the information that needs to be collected to effectively troubleshoot a potential wireless client interoperability issue with a CUWN. This section is intended to serve as a quick reference section, as needed.

I. Problem Definition

- Is the issue restricted to a specific model of access point(s) and/or radio type (2.4 GHz versus 5 GHz)?
- Is the issue observed only on specific version(s) of wireless LAN controller (WLC) software?
- Is the issue experienced with only specific version(s) of client type(s) and/or software (OS version, WLAN driver version,...)
- Are there any other wireless devices which do not experience this issue? If so, what are they?
- Is the issue reproducible while the client is connected to an open SSID, a channel width of 20 MHz, and 802.11ac disabled? (Does the issue happen on 11n mode versus 11ac mode only)
- If the issue is not reproducible with an open SSID, at what minimum security configuration is the issue seen? (PSK or 802.1X on the WLAN)
- What were the previous known-good configuration and software versions?

II. WLC Configuration and Logs

Collect this from the CLI of the WLC(s) in question:

- config paging disable
- show run-config

Alternatively, you can also collect just these output as needed:

- config paging disable
- show run-config no-ap
- show wlan apgroups

Backup of the WLC configuration via TFTP, FTP,...(GUI: **Commands > Upload File > Configuration**)

Syslogs from the WLC

III. Client Device Information

- Client type (tablet, smartphone, notebook PC,...)
- Device make and model
- OS version
- WLAN adapter model
- WLAN adapter driver version
- Supplicant used (Windows Zero Config / Auto Config, Intel PROSet,...)
- Security configured for use by the wireless client and WLAN (Open, PSK, EAP-PEAP/MSCHAPv2,...)

  Note: Any client parameters changed from default settings provided by the vendor in question. (sleep state, roaming parameters, U-APSD,...)

IV. Network Topology Diagram

This includes a representation and/or details with regards to the wireless devices in the network (printers/scanners, WLCs,...)

V. Create a Spreadsheet To Record All Client Issues

Example:

MAC Address	Username	Description of Reported Symptom	Time end user observed symptom	Ping default gateway Y/N	WiFi Signal status (Connected/trying to Connect)	Record ipconf /all (or equivalent)

The goal of this exercise is to help identify a common pattern, and to showcase a more accurate picture of the issue(s) at hand.

VI. Show and Debug Commands on the WLC

Collect these WLC debugs via the CLI:

- **config sessions timeout 0**
- **debug client <MAC_address>**
- **debug dhcp message enable**

Add the additional debugs on case by case basis:

- **debug aaa detail enable** - use this if there are authentication related issues with AAA server
- **debug aaa events enable** - use this if there are authentication related issues with AAA server
- **debug aaa all enable** - use this for auth issues; this is verbose so use it only when needed (for AAA override cases and the like)
- **debug mobility handoff** - use when roaming issues between WLCs

Collect the output for the WLC show commands via the CLI:

- **config paging disable**
- **show time**
- **show client detail <mac-address of client>** (note the client state on the WLC)

- Ping the client from the WLC

Once the test is complete, use this command to stop all current debugs on the WLC:

- **debug disable-all**

VII. Show and Debug Commands on the AP

Lightweight Cisco IOS® APs

This section details the debugs required for the 1700/2700/3700 series or prior model access points.

To avoid an AP session timeout at the time of a Telnet/SSH/console session, use these commands:

- **debug capwap console cli**
- **config t**
- **line console 0** -- use to modify serial session timeout parameters
- **line vty 0 4** -- use to modify Telnet/SSH session timeout parameters
- **exec-timeout 0**
- **session-timeout 0**
- **term len 0**

Before you start the test, collect a sample of these show commands on the AP. At a minimum collect two samples of this output, both before and after completion of tests with the use of these AP show commands via the CLI:

- **term len 0**
- **show clock**
- **show tech**
- **show capwap client mn**
- **show int do1 dfs**
- **show logging**
- **more event.log**
- **show trace dot11_rst display time format local**
- **show trace dot11_rst**
- **show trace dot11_bcn display time format local**
- **show trace dot11_bcn**

Collect these AP debugs via the CLI:

- **debug dot11 { d0 | d1 } monitor addr <MAC_address>**
- **debug dot11 { d0 | d1 } trace print clients mgmt keys rxev txev rcv xmt txfail ba**
- **term mon**

Once the test is complete, use this command to disable the debugs:

- **u all**

AP-COS APs

This section details the debugs required for the 1800/2800/3800 series APs.

To avoid an AP session timeout at the time of a Telnet/SSH/console session, use these commands:

- **exec-timeout 0**

Before you start the test, collect a sample of the show commands on the AP. At a minimum collect two samples of this output, both before and after completion of tests with the use of these AP show commands via the CLI:

- **term len 0**
- **show clock**
- **show tech**
- **show client statistics <client_MAC-address>**
- **show cont nss status**
- **show cont nss stats**
- **show log**

For the 1800 series access points, collect these AP debugs via the CLI:

- **debug dot11 client level events addr <client_MAC-address>**
- **debug dot11 client level errors addr <client_MAC-address>**
- **debug dot11 client level critical addr <client_MAC-address>**
- **debug dot11 client level info addr <client_MAC-address>**
- **debug dot11 client datapath eapol addr <client_MAC-address>**
- **debug dot11 client datapath dhcp addr <client_MAC-address>**
- **debug dot11 client datapath arp addr <client_MAC-address>**
- **term mon**

For the 2800/3800 series access points, collect these AP debugs via the CLI:

- **config ap client-trace address add <client_MAC-address>**
- **config ap client-trace filter all enable**
- **config ap client-trace output console-log enable**
- **config ap client-trace start**
- **term mon**

Once the test is complete, use this command to disable the debugs:

- **config ap client-trace stop**

VIII. Client Side Captures

Collect either a promiscuous Netmon 3.4 (Windows XP or 7 only) or Wireshark packet capture from the client device's WLAN adapter.

IX. OTA Captures

802.11n Captures

- AirPCAP with Wireshark

- MacBook Pro
- OmniPeek Professional, Enterprise,...
- OmniPeek Remote Assistant (ORA)
- Cisco AP in Sniffer mode

802.11ac Captures

- For 11ac 3SS captures, you can use a 2014 Macbook Pro or later running 10.10.x or higher (don't use MacBook Air for 11ac captures if possible, as it is a 2SS only device currently).
- You can also use either a 2702, 3702 or similar Cisco AP in sniffer mode.
- For roaming scenarios and with the use of professional network analysis software such as OmniPeek from Savvius. It is recommended to use multiple, compatible 802.11ac capable USB WLAN adapters, such as the Savvius WiFI Adapter for OmniPeek (802.11ac), Netgear A6210, or similar.

XI. Appendix A - Additional Tips and Tricks

Windows

To collect some additional information with regards to the current wireless connection and other related details directly from a Windows PC. You can make use of these netsh wlan related commands in the Windows command line (CMD):

```
C:\Users\engineer>netsh wlan show ?
These commands are available:
Commands in this context:
show all       - Shows complete wireless device and networks information.
show allowexplicitcreds - Shows the allow shared user credentials settings.
show autoconfig - Shows whether the auto configuration logic is enabled or
                  disabled.
show blockednetworks - Shows the blocked network display settings.
show createalluserprofile - Shows whether everyone is allowed to create all
                  user profiles.
show drivers   - Shows properties of the wireless LAN drivers on the system.
show filters   - Shows the allowed and blocked network list.
show hostednetwork - Show hosted network properties and status.
show interfaces - Shows a list of the wireless LAN interfaces on
                  the system.
show networks  - Shows a list of networks visible on the system.
show onlyUseGPProfilesforAllowedNetworks - Shows the only use GP profiles on GP
configured networks setting.
show profiles  - Shows a list of profiles configured on the system.
show settings  - Shows the global settings of wireless LAN.
show tracing   - Shows whether wireless LAN tracing is enabled or disabled.



C:\Users\engineer>netsh wlan show interfaces
There are 3 interfaces on the system:

    Name                   : Wireless Network Connection 8
    Description            : WildPackets Conceptronic Nano Wireless 150Mbps USB
Adapter #5
    GUID                   : 6beec9b0-9929-4bb4-aef8-0809ce01843e
    Physical address       : c8:d7:19:34:d5:85
    State                  : disconnected
```

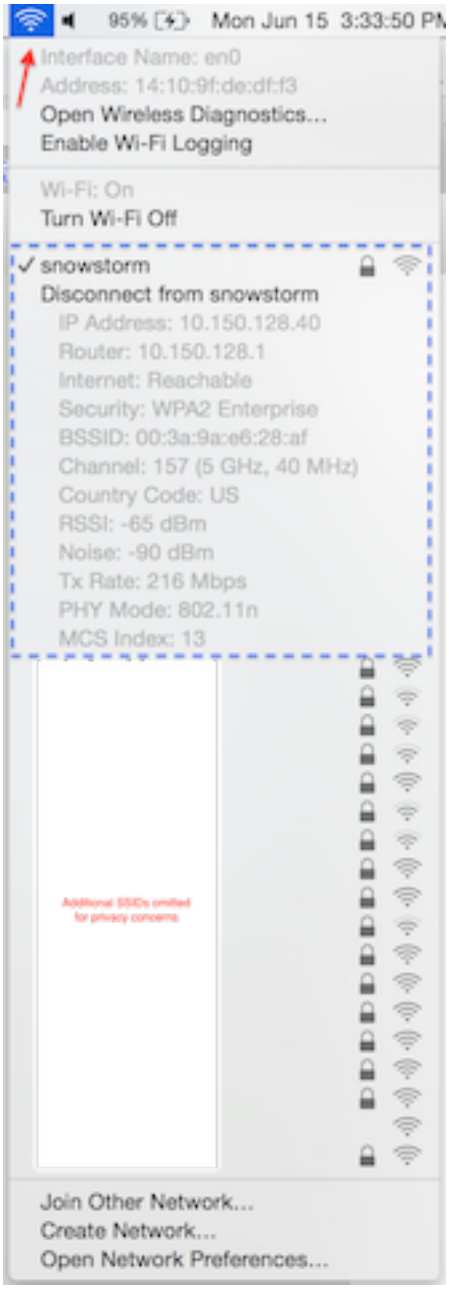
```
    Name                   : Wireless Network Connection 4
    Description            : WildPackets Conceptronic Nano Wireless 150Mbps USB
Adapter
    GUID                   : 23aa09d4-c828-4184-965f-4e30f27ba359
    Physical address       : 48:f8:b3:b7:02:6e
    State                  : disconnected

    Name                   : Wireless Network Connection
    Description            : Intel(R) Centrino(R) Advanced-N 6200 AGN
    GUID                   : 8fa038f8-74e0-4167-98f9-de0943f0096c
    Physical address       : 58:94:6b:3e:a1:d0
    State                  : connected
    SSID                   : snowstorm
    BSSID                  : 00:3a:9a:e6:28:af
    Network type           : Infrastructure
    Radio type             : 802.11n
    Authentication         : WPA2-Enterprise
    Cipher                 : CCMP
    Connection mode        : Profile
    Channel                : 157
    Receive rate (Mbps)    : 300
    Transmit rate (Mbps)   : 300
    Signal                 : 80%
    Profile                : snowstorm

    Hosted network status  : Not started


C:\Users\engineer>netsh wlan show networks bssid | more

Interface name : Wireless Network Connection
There are 21 networks currently visible.

SSID 1 : snowstorm
    Network type            : Infrastructure
    Authentication          : WPA2-Enterprise
    Encryption              : CCMP
    BSSID 1                 : 00:3a:9a:e6:28:af
         Signal             : 99%
         Radio type         : 802.11n
         Channel            : 157
         Basic rates (Mbps) : 24 39 156
         Other rates (Mbps) : 18 19.5 36 48 54
    BSSID 2                 : 00:3a:9a:e6:28:a0
         Signal             : 91%
         Radio type         : 802.11n
         Channel            : 6
         Basic rates (Mbps) : 1 2
         Other rates (Mbps) : 5.5 6 9 11 12 18 24 36 48 54

-- More  --
```

macOS (formerly OS X)

In order to collect the equivalent output as the **ipconfig /all** command on a Windows PC, you can instead use the common Linux/Unix command of **ifconfig** to list detailed information for all of the network interfaces on an Apple MacBook. As needed, you can also specify to receive the output for just the native wireless interface for a given MacBook (either en0 or en1, it depends on the model). Such as this example:

```
bash-3.2$ ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
ether 14:10:9f:de:df:f3
inet6 fe80::1610:9fff:fede:dff3%en0 prefixlen 64 scopeid 0x4
inet 10.150.128.40 netmask 0xffffe000 broadcast 10.150.159.255
nd6 options=1<PERFORMNUD>
media: autoselect
status: active
```

In order to get some quick but detailed information with regards to the current wireless connection on a MacBook. You can also select the WiFi icon in the top right corner of the desktop while you simultaneously hold the **option** button on your keyboard as shown in the image.



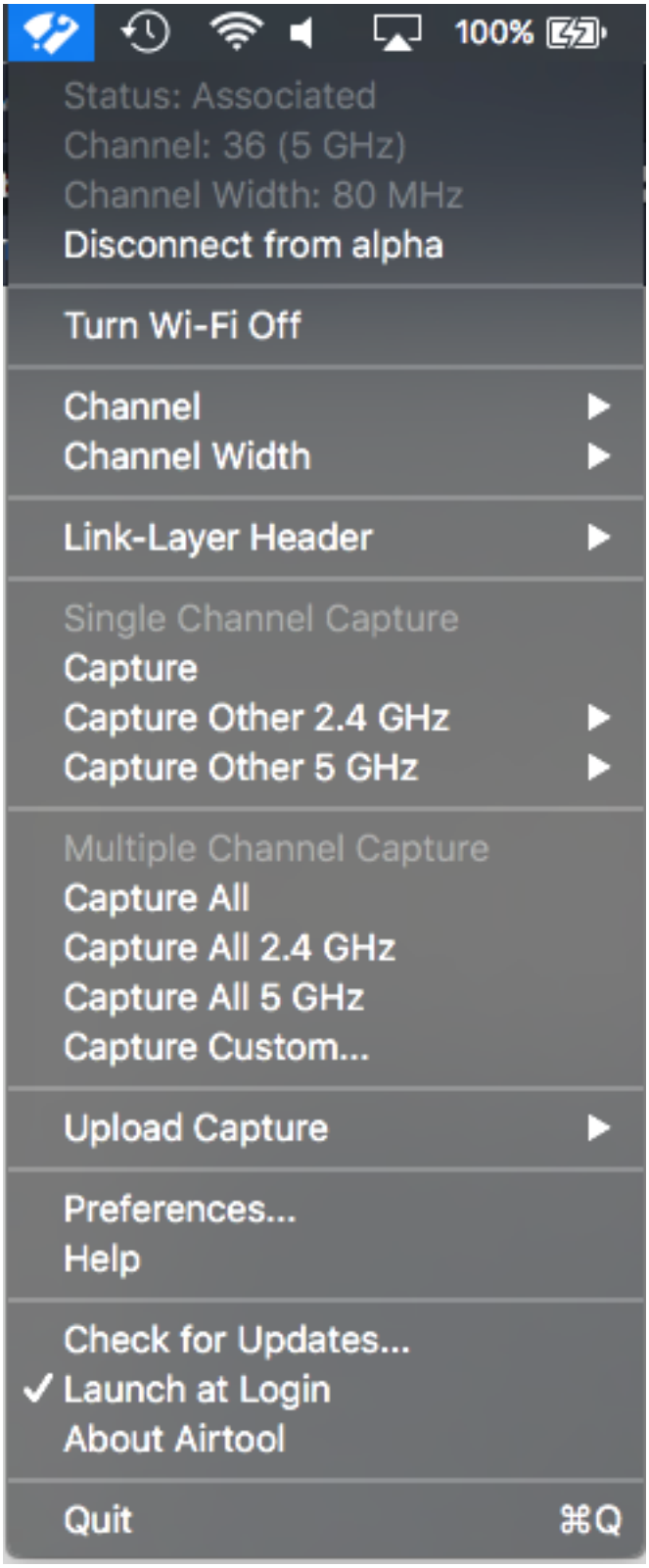
Another useful option is to utilize the hidden command line utility called airport. It is highly recommended to only utilize this with your own MacBook or one in use in a lab environment. As some network administrators might not wish to grant access to this utility on an end user's MacBook, so use the appropriate level of caution accordingly. To proceed, enter this in Terminal on the MacBook in question:

```
sudo ln -s
/System/Library/PrivateFrameworks/Apple80211.framework/Versions/Current/Resources/airport
/usr/local/bin/airport
```

Now you can call upon the airport CLI utility with ease. An example of which includes this:

```
bash-3.2$ airport -I
     agrCtlRSSI: -61
     agrExtRSSI: 0
    agrCtlNoise: -90
    agrExtNoise: 0
          state: running
        op mode: station
     lastTxRate: 216
        maxRate: 300
lastAssocStatus: 0
    802.11 auth: open
      link auth: wpa2
          BSSID: 0:3a:9a:e6:28:af
           SSID: snowstorm
            MCS: 13
        channel: 157,1
```

To further ease the process to collect a reliable, single 802.11 channel OTA packet capture with the use of the capabilities of a MacBook Pro or similar. You can either leverage the embeded capabilities in macOS with the use of the Wireless Diagnostics > Sniffer method or similar as discussed previously, but optionally you can use a third-party utility called Airtool as well (OS X 10.8 and later). The benefit is a simple interface to quickly collect an OTA packet capture, which gets saved directly to the desktop with just a few clicks through the app UI right from the top menu bar on your screen.



Further information and download links for Airtool can be found at this URL:

https://www.adriangranados.com/apps/airtool