July 17, 2015

Windows DUT Porting Guidelines

Version 1.0 Beta

11n/WMM/WPA2, VHT, WMM-PS, P2P, PMF, TDLS, VE, WMM-AC

Change History

Version	**Date released**	**Summary**
1.0 Beta	07/17/2015	Beta release

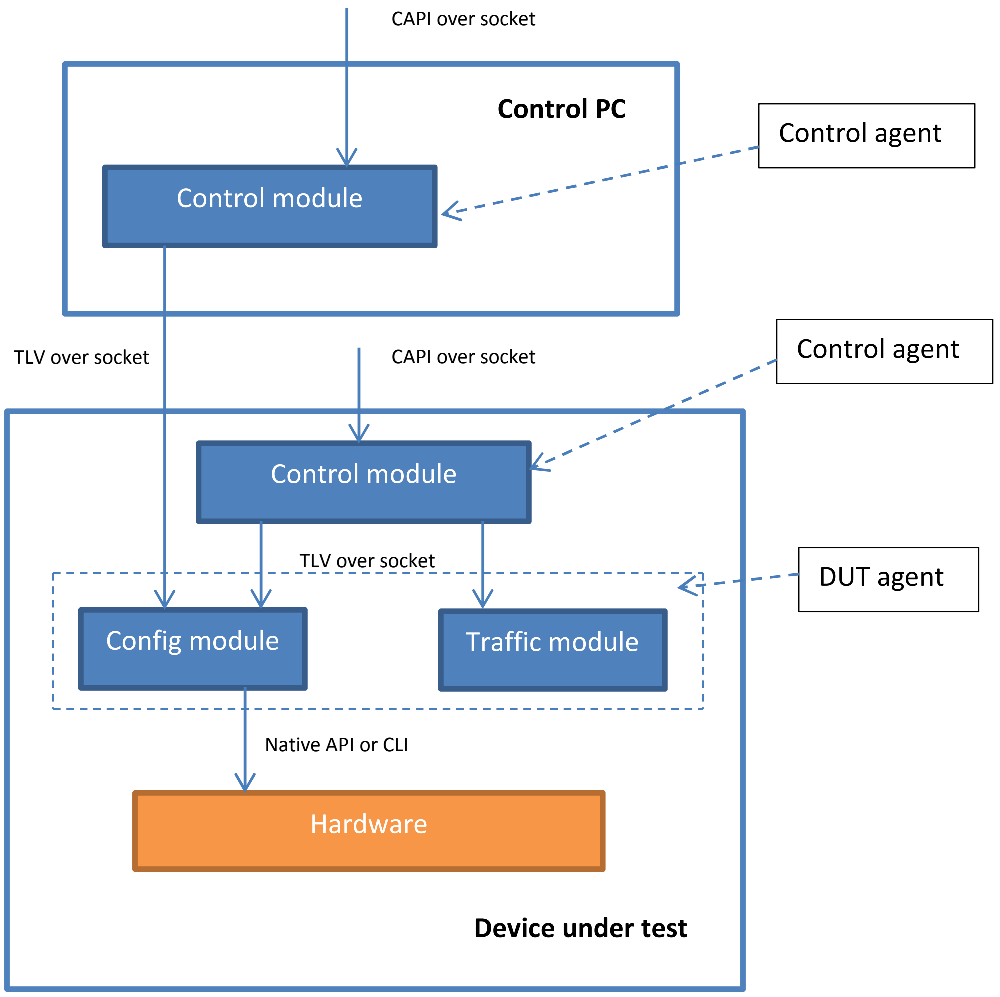
Table of Contents

1. Scope

This document provides guidance for Wi-Fi Alliance members and their software engineers who are responsible for implementing DUT code on Windows platform so that their devices can be tested and certified for Wi-Fi certification programs. It is also reckon as a baseline for open source community to extend functionality and improve quality.

The DUT code consists of two agents (control agent and DUT agent) with several modules, i.e., control module, configuration module, and traffic module. Currently it supports Ethernet connection between the device and test console. Other interfaces can be incorporated as part of future enhancement.

2. Overview of DUT code architecture



The Wi-Fi Alliance provides the sample Windows DUT source code which includes:

1. Control module in control agent: The code receives the CAPI command from test console and converts it to the struct which is then wrapped into TLV format and sent to DUT agent. The control agent can reside on either a control PC which is detached from underlying device or actually the device. The communication interface between control agent and DUT agent in the sample code is TCP socket over Ethernet. Other interfaces can be developed to support a variety of connections such as USB, USB over serial, serial or wireless and so on.
2. Config module in DUT agent: The agent receives data packet in TLV format from network over Ethernet interface. It will then invoke the function corresponding to the decoded command based on its tag value. The function will in turn call either the low level API or CLI batch to execute the command.
3. Traffic module in DUT agent: The agent will generate the UDP based traffic and start sending/receiving it to/from the peers. The data packets generated are at a either fixed rate or maximized rate which the underlying OS can support.

3. Working with the source code

The overall source code structure comprises the following four subdirectories:

Directory name	**Description**
/win_ca	Contains files related to control agent communication with DUT agent and test console
/win_dut	Contains files relate to DUT agent initialization and communication with control agent
/win_inc	Contains header files shared by control and dut agent
/win_lib	Contains various library files shared by control and dut agent
/Tools/myping	Contains myping utility source files

The win_ca and win_dut directory contains the visual studio project file for building win_ca.exe and win_dut.exe binary, respectively.

The win_ca directory includes the following source files:

Source file name	**Description**
/win_ca/win_ca.h	Defines the global structures used by control agent.
/win_ca/win_ca.c	The main entry of control agent.

The win_dut directory includes the following source files:

Source file name	**Description**
/win_dut/win_dut.h	Defines the global structures and variables used by

	dut agent.
/win_dut/win_dut.c	The main entry of dut agent.
/win_dut/win_dut_init.c	Various buffer initializations.

The win_inc directory includes the following source files

Source file name	Description
/iphdr.h	Various protocol header definitions used by the raw socket.
/resolve.h	Contains common name resolution and name printing routines.
/wfa_ca_resp.h	Contains the declaration of functions which return command execution results to test console.
/wfa_cmds.h	The definition of command types.
/wfa_debug.h	The declaration of functions for debugging and logging.
/wfa_main.h	Contains the common headers used by control agent and dut agent.
/wfa_miscs.h	Contains the declaration of miscellaneous functions.
/wfa_rsp.h	Defines the data structures which store the command response information.
/wfa_sock.h	The socket function declaration.
/wfa_tg.h	Defines the macros and structs used for traffic generation.
/wfa_tlv.h	The definition of TLV tag and its encoding/decoding routines.
/wfa_types.h	Defines the string and buffer length macros.
/wfa_ver.h	Defines the system version of dut agent.
/wfa_wmmps.h	Macros and structs definition, and function declaration for WMM-PS

The win_lib directory contains the following source files:

Source file name	Description
/wfa_ca_resp.c	Contains the functions which return results to test console
/wfa_cmdproc.c	The functions which parse CAPI commands
/wfa_cmdtbl.c	The declaration of functions for implementing CAPI commands.
/wfa_cs.c	The function implementations of CAPI commands.
/wfa_miscs.c	Contains the miscellaneous utility functions.
/wfa_sock.c	C wrappers of winsock.
/wfa_tg.c	Contains traffic-related CAPI command processing functions
/wfa_thr.c	Thread function for sending/receiving packets based on the priority of packets and the affiliated

	functions which create QoS flow using TC APIs.
/wfa_tlv.c	TLV tag encoding and decoding routines.
/wfa_typestr.c	The declaration of CAPI parsing functions in control agent.
/wfa_wmm_tc.c	Contains the functions used for the programs where the user priority needs to be set for the outgoing packets.
/wfa_wmmps.c	The functions used exclusively for WMM-PS program.

The Tools directory contains the following source files:

Source file name	**Description**
/Tools/myping/iphdr.h	Various protocol header definitions used by the raw socket.
/Tools/myping/resolve.h	The declaration of network address resolution functions.
/Tools/myping/resolve.c	Contains functions for print, formatting and resolving the specified address.
/Tools/myping/myping.c	Contains functions for how to use raw sockets to send ICMP requests and receive response.

4. Overview of porting process

The Device under Test (DUT) is a device usually with the following two network interfaces:

- The Wi-Fi Interface is typically implemented via a Wi-Fi chipset within the DUT which provides 802.11x MAC and PHY functionality together with the driver and other software.
- The control interface is typically an Ethernet link or others to which the DUT can be connected and receives CAPI commands.

The sample Windows DUT code is designed with easy portability and modularity in mind. Conceptually it consists of the following three functions:

- Communication: The CAPI commands are received from and the response is generated and sent back to test console via the TCP control link.
- Device configuration and action: The command and associated parameters are used to configure the device in preparation for a given test or to execute the specified action.
- Traffic generation: The data packets are generated and sent over the air through the wireless interface as per the parameters specified (payload size, framerate, and IP address and port etc.).

OS dependent APIs:

The OS dependency includes the use of socket and thread, and other system APIs with regard to QoS and system time. Although these APIs have been tested on Windows 7, 8 and 10 platforms, when porting to other Windows platforms, it should be aware of the existence of dependency. The following table shows the distribution of these APIs in different source files.

Windows API	**Files**
socket	Mainly in wfa_sock.c, select() is used in wfa_dut.c, wfa_thr.c and wfa_wmmps.c
Thread/mutex/condition variable	wfa_dut.c, wfa_thr.c, wfa_tg.c, wfa_wmm_tc.c, and wfa_wmmps.c
QoS TC API	wfa_thr.c
System time	wfa_dut.c and wfa_miscs.c
Registry	wfa_cs.c

Configuration and action APIs:

These APIs are mainly located in the wfa_cs.c file and invoked by the main code to configure the device and execute the Wi-Fi specific actions.

These APIs can in turn invoke vendor's low level APIs which will manage and control the device via the device driver. Each function in the wfa_cs.c file corresponds to a CAPI command. Some implementations are achieved through Windows netsh utility.

Another way of implementing CAPI command is through CLI. Here are the steps to follow for the implementation:

1. Add the CAPI command to be implemented to the wfa_cli.txt file according to the following format:
   - <CAPI command>-TRUE,
   - <CAPI command>-FALSE,

   Note: TRUE means the command needs to return the result to test console. FALSE means return null.

2. Create a batch file with exactly the same file name as CAPI command, provide the implementation in it and write the result back to windows DUT using setx. Each CLI must return an error code status through the environment variable - WFA_CLI_STATUS.

   WFA_CLI_STATUS Code Definition:
   - WFA_CLI_COMPLETE(0)

- WFA_CLI_ERROR(1)
- WFA_CLI_INVALID(2)

If there is any return value from CLI, it should be returned through the environment variable WFA_CLI_RETURN (String). The return value format is <var1_name, var1_value, var2_name, var2_value>. For example, if CLI "STA_P2P_START_GROUP_FORMATION" returns two values "result" and "groupid" then enviorment variable WFA_CLI_RETURN should be set to "result,client,groupid,00:11:22:33:44:55 DIRECT-S2".

The setx or other approaches can be used to write the result back to Windows environment variables. For example,

- SETX WFA_CLI_STAUS 0
- SETX WFA_CLI_RETURN "result,client,groupid,00:11:22:33:44:55 DIRECT-S2"

5. API implementation for new command

When implementing a new CAPI command, a simpler approach is to follow the steps above and do it in the CLI way. If using API is preferred, the following is the suggested procedure:

1. Define the new command's TLV tag in wfa_tlv.h.
2. Declare the response function in wfaf_ca_resp.h and add the function implementation in wfa_ca_resp.c.
3. Add a command parsing handling function to the function array of nameStr in wfa_typestr.c and implement it in wfa_cmdproc.c.
4. Define new struct for the command in wfa_cmds.h if it is different from dutCommand_t and a response struct in wfa_rsp.h if it is not a generic one as dutCmdResponse_t.
5. Declare the function in gWfaCmdFuncTbl of wfaf_cmdtbl.c and Implement it in wfa_cs.c.

6. Program specific build options

The source code defines a few program specific compiling switches for building the executable for either individual program or all the programs. The relationship among the switches is illustrated as below:

Build options	**Programs to build**
WFA_WMM_WPA2	11n, VHT, TDLS
WFA_WMM_PS	WMM-PS
WFA_WMM_AC	WMM-AC
WFA_P2P	P2P
WFA_WMM_WPA2 and WFA_VOICE_EXT	Voice Enterprise
Not specified	PMF, WFD, WFDS, HS2
All specified	All the programs as state above