**Abstract**

The WiFi Metrics Test Suite adds an automated framework for configuring, running and monitoring the tests used by WDC Networking Test for verifying the quality of the WiFi sub-system. It also provides mechanisms for objectively measuring quality and tracking the quality metric as modifications are applied.

|  |  |  |
| --- | --- | --- |
| ***Owner*** | ***Revision Notes*** | ***Date*** |
| [micedm@microsoft.com](mailto:micedm@microsoft.com) | Document Created | 4/15/2007 |
| [vinayv@microsoft.com](mailto:vinayv@microsoft.com) | Updated and Included Appendix D – AS Setup | 4/15/2008 |

**Table of Contents**

[Introduction 4](#_Toc268276276)

[Objective 4](#_Toc268276277)

[Methodology 4](#_Toc268276278)

[Quality Metrics 6](#_Toc268276279)

[Hardware Components 8](#_Toc268276280)

[Software Components 10](#_Toc268276281)

[Logical Architecture 10](#_Toc268276282)

[Physical Architecture 12](#_Toc268276283)

[Configuration 14](#_Toc268276284)

[Configuration Files 14](#_Toc268276285)

[XML Syntax 14](#_Toc268276286)

[Overall Test Configuration 15](#_Toc268276287)

[Test-Suite Configuration 17](#_Toc268276288)

[Configuration Utilities 19](#_Toc268276289)

[APCTool – AP-Control Client 19](#_Toc268276290)

[WiFiTool – WiFi Subsystem Configuration Utility 21](#_Toc268276291)

[AP-Control Server Configuration 22](#_Toc268276292)

[Authentication Server Configuration 24](#_Toc268276293)

[NDT Server Configuration 25](#_Toc268276294)

[NDIS Perf Server Configuration 26](#_Toc268276295)

[Finding Names for a Network Interface 26](#_Toc268276296)

[Echo Service Configuration 27](#_Toc268276297)

[Ad Hoc Station Configuration 28](#_Toc268276298)

[Prerequisites for Running the WiFi Metrics Test 28](#_Toc268276299)

[Command-Line Parameters for the WiFi Metrics Test 30](#_Toc268276300)

[Metrics Reporting and Storage 32](#_Toc268276301)

[WiFi Metrics Tests 33](#_Toc268276302)

[Common Test Parameters 33](#_Toc268276303)

[Directory Organization 33](#_Toc268276304)

[Default Run-Mode Parameters 33](#_Toc268276305)

[Device-Under-Test Configuration 34](#_Toc268276306)

[Communication Parameters 34](#_Toc268276307)

[Access Point Configuration 36](#_Toc268276308)

[Authentication Certificate Enrollment 38](#_Toc268276309)

[EAP Authentication / Login Credentials 39](#_Toc268276310)

[Stability Tests 40](#_Toc268276311)

[1001-1050: Ndt\_1card – NDIS One-Card Test 40](#_Toc268276312)

[Communication Tests 43](#_Toc268276313)

[1101-1150: Ndt\_2card – NDIS Two-Card Test 43](#_Toc268276314)

[Security Tests 45](#_Toc268276315)

[2000-15000: AuthMatrix – Authentication Matrix Test 45](#_Toc268276316)

[Appendix A: Supported Access Points 50](#_Toc268276317)

[Appendix B: Supported RF Attenuators 52](#_Toc268276318)

[Appendix C: Example Hardware Bill of Materials 53](#_Toc268276319)

[RF Attenuation 53](#_Toc268276320)

[RF Isolation Enclosures (2) 54](#_Toc268276321)

[Access Points and Test Network 55](#_Toc268276322)

[Appendix D: Authentication Server Setup 56](#_Toc268276323)

[Overview 56](#_Toc268276324)

[Pre-requisites 56](#_Toc268276325)

[Install and Configure Active Directory and DNS Server 59](#_Toc268276326)

[Install and Configure DHCP 72](#_Toc268276327)

[Installing Internet Information Services (IIS) 82](#_Toc268276328)

[Setting up the Internet Authentication Service (IAS). 86](#_Toc268276329)

[Setting up RADIUS clients 89](#_Toc268276330)

[Installing and Configuring Certificate Services 91](#_Toc268276331)

[Enabling Certificate Templates 96](#_Toc268276332)

[Setting up user groups and accounts. 99](#_Toc268276333)

[Setting Remote Access Policies in IAS 104](#_Toc268276334)

# Introduction

## Objective

As of early 2007 these are the primary existing and proposed automated tests in use within WDC Core Networking Test for verifying the quality of WiFi adapters:

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Description*** | ***Notes[[1]](#footnote-1)*** |
| ndt\_1c | NDIS One-Card Miniport Driver Test | ATCL |
| ndt\_2c | NDIS Two-Card Miniport Driver Test | ATCL |
| ndt\_wlan\_1c | NDIS One-Card WLAN Driver Test | ATC |
| ndt\_wlan\_2c | NDIS Two-Card WLAN Driver Test | ATC |
| ndis\_perf | NDIS Driver Performance Test | ATCL |
| AuthMatrix | Authentication Matrix Test Suite | ATCL |
| WiFiRoam | WiFi Roaming Performance Test Suite | M |
| WiFiPower | WiFi Power Usage Measurement Test | MC |
| VoIPReady | WiFi VoIP Ready Performance Test | P |

Each of these tests measures different aspects of the reliability and/or performance of the WiFi sub-system. While they each reports important results, they lack:

* a mechanism for objectively measuring and reporting the overall quality of the WiFi subsystem, and
* an easily configured and managed test framework.

The WiFi Metrics project will solve those problems by:

* defining a standardized hardware and software configuration which will run all the tests with minimal operator interaction, and
* building a control system which will automatically run all the tests, monitor and consolidate the results, report them to the operator and register them in a historical database.

## Methodology

Almost all of the existing automated tests are constructed around the Tux test framework. This allows the tests to be remotely initiated and monitored from a central server. We will capitalize on this capability to build a Test Server which automates all the configuration, management and monitoring of the tests. As each test finishes, the results will be communicated back to a central server to be stored in a historical database, summarized and reported on the WiFi Metrics web-site.

The most important current exceptions to the Tux pattern are WiFiRoam and WiFiPower. The WiFiRoam test is currently a long-term series of automated inter-AP, USB and cellular roams which move a device’s connection through a normal “day-in-the-life” scenario over and over. The test results are displayed on the device’s UI and not currently available for external access. This will be resolved by adapting the test to the standard Tux pattern.

Similarly, the WiFiPower test is normally a manual process where the operator connects the device’s battery to a Power Monitor and records the power usage while a test application runs the device through a series of WiFi operations. Currently, the power usage results are manually calculated by the operator by analyzing the log of the test and associating the log with the power readings. This process will be automated and the results will be stored in a file which can be retrieved by the Test Server.

# Quality Metrics

There are many existing and planned tests for WiFi, but they all fall within a relatively small number of categories. Each of these categories will be considered a separate metric. Once all the test categories are completed, they will be accumulated into an overall quality metric for the device.

This is the initial list of test categories and their estimated run times:

|  |  |  |
| --- | --- | --- |
| ***Metric Name*** | ***What’s Tested?*** | ***Approx Duration*** |
| Stability | Confirms standard NDIS and WLAN functionality and data stability. (Was NDIS One-Card and One-Card-WLAN.) | 1 hour |
| Communication | Confirms standard NDIS/WLAN functionality with external NIC (Was NDIS Two-Card and Two-Card-WLAN.) | 1 hour |
| Performance | Confirms throughput numbers for peek and sustained. (Was NDIS Perf.) | 1/2 hour |
| Security | Confirms authentication and security functions by connecting using all supported security modes and keys. | 2 hours |
| Roaming/Stress | Measures and stresses AP-to-AP roaming. (Was manual MTTF test.) | 8 hours |
| Power | Measures the power utilized by WiFi in various states from idle to high-speed data-communication. | 3 hours |
| VoIP | Confirms functionality such as QoS, WMM and Latency/Jitter required for proper VoIP operation | 1 hour |
| Miscellaneous | Other functional tests of features such as Ad Hoc, PMK caching, proper SSID handling and so on | 2 hours |

The first three test categories are low-level tests of the basic capabilities and performance of the WiFi driver:

* The **Data Stability** tests were originally known as the NDIS One-Card and One-Card-WLAN Miniport Driver Test. These tests assess the functionality of a miniport driver for a single network card without communicating outside the device being tested.
* The **Communication** tests were originally known as the NDIS Two-Card and Two-Card-WLAN Network Card Miniport Driver Test. These tests assess the ability of a miniport driver to communicate reliably with another Access Point or Ad Hoc network.
* The **Performance** tests were originally known as the NDIS Performance Miniport Driver Test. These tests move data through WiFi connections as fast as possible to assess the miniport driver’s performance.
* The **Security** tests were originally known as the WiFi Authentication Matrix Test Suite. These tests assess the ability of the target device to connect and communicate using all the wireless authentication and encryption modes supported by the wireless NIC.
  + Authentication modes supported: Open, Shared, WPA-EAP, WPA-PSK, WPA2-EAP and WPA2-PSK
  + Encryption modes supported: WEP, TKIP, AES and clear-text (no encryption)
  + EAP authentication modes supported: TLS and PEAP
  + Plus various key compositions for each of the shared-key security modes.
* The **Roaming and Stress** test repeatedly roams the target device through a series of Access Points to measure roaming speed and assess its ability to handle a “real-world” usage scenario for a long duration. The extent of this test is designed to simulate approximately three months of regular, continuous, usage.
* During the **Power** consumption test, the device is connected to an automated power monitor and WiFi is configured for various usage scenarios from idle to high-speed data-communication. The power monitor automatically calculates and records the power usage for each of these scenarios and the averages are compared against the limits imposed by the test.
* The **VoIP** Specific tests are designed to test aspects such as QoS (Quality of Service) WMM (Wireless Multi-Media) and Latency/Jitter to verify the device will perform adequately to support VoIP-over-WiFi operation.
* Finally the **Miscellaneous** tests verify operation of the second-tier WiFi functions such as PMK Caching, Ad Hoc communication, Hidden and random SSIDs and so on. Many of the remaining manual tests are being retired and moved into this category.

# Hardware Components

At a high level, these are the WiFi Metrics hardware components:

**Test Control Server**: Runs applications which control the configuration of the attached Access Points, RF Attenuators and USB Switches (if any).

**Radius and DHCP Server**: Provides authentication services and issues IP addresses during the tests. As indicated, this need not be a separate server, but can run on the Test Control Server system.

**WiFi Metrics Database and Web Site**: This optional server stores historical test-result records and manages a web-site for viewing the summarized results.

**Device Under Test (DUT)**: Runs the WiFi Metrics test software. Can be connected to the Test Server using KITL. If so, the test progress can be monitored from a Platform Builder IDE on the Server.

**NDTServer Support Device:** Serves as the Support Device for the [NDIS Two-Card Tests](#_Communication_Tests). Runs the [NDT Server](#_NDT_Server_Configuration) to confirm the DUT’s ability to connect to and communicate with a second device. This can be any kind of device with a WiFi connection. The illustration shows a mobile device controlled by a Platform Builder via a KITL connection.

**WiFi Access Points**: Must be chosen from the list in [Appendix A](#_Appendix_A:_Supported) of devices which can be automatically configured by the AP-Control Server.

[**RF Isolation Enclosure(s)**](#_RF_Isolation_Enclosures)**:** Isolates the Access Points so their RF output can be totally controlled by the RF Attenuator.

[**RF Attenuator**](#_RF_Attenuation): Controls the radiated output from the Access Points. Like those, this must be chosen from the list in [Appendix B](#_Appendix_B:_Supported) of devices which can be automatically configured by the AP-Control Server.

**USB Switches**: Automatically enables or disables a USB connection between the Test Server and the DUT. This is used to simulate the “cradling” of the device so it can connect using Desktop Passthru or ActiveSync. Alternatively, if the DUT has no other network connection, this device can be used to connect it to the Test Server to initiate tests and retrieve results.

# Software Components

## Logical Architecture

****

**WQMTestManager**: Manages all phases of a WiFiMetrics test. Parses configuration files to determine which tests to run and high-level configuration items like test name and pass/fail criteria. Usually this configuration file includes pointers to a series of lower-level files which describe each test suite in detail. Each of these files supply a complete list of the suite’s test cases and all the information required to configure, start and stop the tests.

The Manager is implemented as a standard Tux DLL. Hence, it fits naturally into the CETK, LTK and WTT frameworks. The initial job of each Tux DLL is the creation of the Tux function table. This gives Tux the list of all the test cases in the DLL. The Manager generates this table by parsing the configuration files. When Tux calls the individual test functions, the Manager intercepts the calls, loads the appropriate test DLL and hands off to the low-level test function.

**WiFiBase:** All recent WiFi tests are written as classes derived from this base class. To derive from WiFiBase, each test class is must define the Init(), Run() and Cleanup() virtual methods. In addition, each class defines a list of the configuration objects it needs. This list is passed to the Manager to be initialized from information in the configuration files.

**WQMTestSuiteBase:** This Factory base class defines a “test-suite” pattern for WiFiMetrics. Each class derived from this base defines an independent DLL and wraps one or more WiFiBase-derived test classes to provide all the initialization and cleanup processing required by the tests and the Manager.

The example in the diagram above illustrates the interaction of the TestSuiteBase and WiFiBase classes. In the example, SSIDTestSuite specializes WQMTestSuiteBase and BroadcastSSID and RandomSSID specialize WiFiBase. SSIDTestSuite uses BroadcastSSID::GetCmdArgList() and RandomSSID::GetCmdArgList() to tell the Manager how to configure the tests. When the Manager, in turn, calls SSIDTestSuite::Execute() to run a test, the Suite instantiates an instance of either BroadcastSSID or RandomSSID and calls its Init(), Run() and Cleanup() methods.

**WQMTuxTestSuite:** This class specializes WQMTestSuiteBase to wrap and execute legacy Tux DLLs. During initialization it loads the legacy DLL, configures it using its command-line interface and reads its Tux function table. When the Manager calls Execute() to run a test the Suite looks up the test and calls the legacy DLL’s test function.

The standard WQMTuxTestSuite class is sufficient for most legacy Tux DLLs. For DLLs requiring complex initialization or cleanup steps, WQMTuxTestSuite can be specialized to create a new class which automates these steps.

## Physical Architecture



**WiFi Metrics:** The test manager, WiFiMetrics.dll, is a standard Tux DLLs. It accepts and optional command-line parameter supplying the name of the top-level configuration file.

[**Configuration Files**](#_Configuration_Files)**:** When started WiFiMetrics.dll loads its configuration from a single file named, by default, WiFiMetrics.xml. This file tells WiFiMetrics how to navigate the local directory structure then includes a series of lower-level files defining, first, the detailed configuration of the test system and, next, the detailed configuration of each of the test “suites” to be performed.

The information associated with each of the test suites includes the type of suite (whether it is a legacy or a Factory test), the name of the test DLL and the DLL’s entry-point. In addition, the information contains a detailed list of all the test-cases to be run by the suite.

The list of test cases to be run need not contain all the tests defined in the test suite. WiFiMetrics will only run the listed cases. This provides a mechanism for quickly defining and running a sub-set of the test cases for special purposes. As a special case of this, certain test cases can be marked as “BVT” (Basic Viability Tests) and, if the top-level “BvtTestRun” flag is set, only those tests will be run.

For more information see the [Configuration](#_Configuration) section later in this document.

[**Legacy Tux Interface**](#_Logical_Architecture:)**:** This interface, WQMTuxTestSuite, is described in the [Logical architecture](#_Logical_Architecture:) section. It dynamically loads the legacy DLL and provides methods which simulate a standard Factory DLL.

[**Factory Tests**](#_Logical_Architecture)**:** These test suites each reside in a separate DLL. Unlike a regular Tux DLL, however, the external interface does not run the tests. Instead, the interface provides access to a “Factory” object. The Factory is responsible for creating and running test cases based on the configuration supplied by the test-manager. This makes it possible to generate new test cases by just changing the configuration.

[**Utilities**](#_Configuration_Utilities)**:** Each test suite definition may optionally contain a “<Config>” section which lists the external utilities to be run before the tests. This allows each suite to configure the Access Points and connections necessary for its tests to run properly. Two new utilities, [APCTool](#_APCTool) and [WiFiTool](#_WiFiTool), come with WiFiMetrics. These utilities (described more fully in the [Configuration Utilities](#_Configuration_Utilities) section) manage the configuration of all the Access Points and WiFi adapters.

[**Log and Summary Files**](#_Metrics_Reporting_and)**:** As a standard Tux DLL WiFiMetrics supports all the normal logging mechanisms. Normally, we use the “-f {file-name}” option to send all the detail logs to a file.

In addition, WiFiMetrics produces a file which summarizes the results for each test suite. In addition to the total pass/fail counts, the entries contain a calculation of the resulting percentage for each suite. Finally, the file includes a calculation of the overall weighted average percentage.

# Configuration

## Configuration Files

### XML Syntax

Almost all the configuration of WiFi Metrics is defined in XML files. In addition to defining test suites and cases, these files define command lines, IP addresses, pass/fail criteria and so on.

The WiFi Metrics parser adds two extensions to the standard XML language, include files and parameter substitution. Both can be seen in this excerpt from the top-level WiFiMetrics.xml file:

|  |
| --- |
| ScriptDirName = "\release\WQMConfigs"    …  <Include  skipFile = "false"  TstCaseIDStart = "1001"  TstCaseIDEnd = "1050"  includeFile = **"$(ScriptDirName)\ndt\_1c\_test.xml**"  /> |

This tells WiFi Metrics to include the file “ndt\_1c\_test.xml”. The “$(ScriptDirName)” string preceding the file-name will be replaced with the ScriptDirName parameter defined earlier. This will result in the following include file name:

includeFile = “\release\WQMConfigs\ndt\_1c\_test.xml"

This simplifies configuration file maintenance by “centralizing” the common definitions. For example, if the configuration files are moved to another directory, the single ScriptDirName definition in the top-level WiFiMetrics.xml files is all that needs to be changed.

After parameter substitution the entire <Include> section will be replaced by the contents of the named file. The resulting combination will be parsed as if the files were one.

This allows the configuration to be split into separate, relatively simple, files and means very few files need ever be modified. In particular, this allows most of the complicated test-case configuration to be isolated in the test-suite description files. By providing these as separate files they will be changed very rarely.

### Overall Test Configuration

#### WiFiMetrics.xml

|  |
| --- |
| <TestSetup name="Directory organization"  RootDirName = "\release"  ScriptDirName = "\release\WQMConfigs"  TempDirName = "\windows"  />  <Include type = "params" includeFile = "$(ScriptDirName)\DUTSetup.xml" />  <Include type = "params" includeFile = "$(ScriptDirName)\HdweSetup.xml" />  <TestGroup  TestGroupName = “Stability”  TestPassCreteria = "100"  TestFailCreteria = "80"  TestGroupWeight = "1.5"  >  <Include  skipFile = "false"  TstCaseIDStart = "1001"  TstCaseIDEnd = "1050"  includeFile = "$(ScriptDirName)\ndt\_1c\_test.xml"  />  </TestGroup> |

This is the primary WiFi Metrics configuration file. It defines up a few basic parameters then includes sub-files which define the rest of the test environment ([DutSetup.xml](#_DUTSetup.xml) and [HdweSetup.xml](#_HdweSetup.xml)) and the test(s) to be performed (ndt\_1c\_test.xml in the excerpt above).

#### DUTSetup.xml

|  |
| --- |
| <TestSetup name="DUT configuration"  **AdapterName** = "ISLP21"  FakeAdapterName = "XWIFI11B1"  ConnectionTimeLimit = "140"  NDTAdapterUnbind = "-nounbind"  /> |

This file describes the device being tested. One primary example from the exceprt above is AdapterName, the name of the WiFi adapter (in this case “ISLP21”) which varies from device to device.

In general, once a test environment is set up (see [HdweSetup.xml](#_HdweSetup.xml) below), this will be the only file which ever needs to be changed.

#### HdweSetup.xml

|  |
| --- |
| <TestSetup name="Communication configuration"  NDTServerHost = "10.10.0.102"  LANServerHost = "172.26.137.101"  LANServerPort = "33331"  LANServerHostArg = "-lHost $(LANServerHost)"  LANServerPortArg = "-lPort $(LANServerPort)"  WiFiServerHost = "10.10.0.10"  WiFiServerPort = "33331"  WiFiServerHostArg = "-wHost $(WiFiServerHost)"  WiFiServerPortArg = "-wPort $(WiFiServerPort)"  xxWiFiServerSSID = "WIFI\_OPEN"  WiFiServerAuth = "Open"  WiFiServerEncr = "ClearText"  WiFiServerAuthArg = "-wAuth $(WiFiServerAuth)"  xxWiFiServerSSIDArg = "-wSSID $(WiFiServerSSID)"  WiFiServerEncrArg = "-wEncr $(WiFiServerEncr)" |

The file describes the local test environment. The excerpt above defines IP addresses for the NDT Server and AP-Control Server. The final “WiFiServer” section describes how the clients should contact the AP-Control Server if they have no other LAN connection.

The clients will use either a fixed LAN or a dynamic WiFi connection to reach the AP-Control Server. The clients determine which to use based on the LANServerHost and WiFiServerSSID parameters. In the example about the definition of the LANServerHost parameter indicates the device supports a fixed LAN connection.

Note that only one of the two parameters can be defined. Since the LAN parameter has been defined the WiFiServerSSID parameter must be removed. To do so, that entry has been “Xed” out. This technique keeps unused parameter entries in the file for later reference.

#### APConfig.xml

|  |
| --- |
| <APController name="ConfigAP1">  <RFAttenuatorState>  <Current> 50 </Current>  </RFAttenuatorState>  <AccessPointState>  <Ssid> ROAM\_HOMEAP </Ssid>  <RadioState> on </RadioState>  <SsidBroadcast> off </SsidBroadcast>  <Authentication> Open </Authentication>  <Cipher> WEP </Cipher>  <WEPKeys activeKey="1">  <WEPKey index="1"> 01.23.45.67.89 </WEPKey>  </WEPKeys>  <RadiusServer> 10.10.0.1 </RadiusServer>  <RadiusPort> 1812 </RadiusPort>  <RadiusSecret> 0123456789 </RadiusSecret>  </AccessPointState>  </APController> |

This file contains a series of APController definitions. Each of these definitions detail how one of the Access Point and its associate RF-Attenuator should be configured. The file is sent to the AP-Control Server by the [APCTool](#_APCTool) utility. The Server interprets the file and, if necessary, reconfigures the devices to match.

The RFAttenuatorState section in the example above indicates the RF-Attenuator should be set to attenuate the Access Point’s RF output by 50db.

The AccessPointState section contains settings indicating the Access Point’s SSID should be set to ROAM\_HOMEAP, its radio should be turned on and it should not broadcast its SSID. Next, the Access Point’s authentication is set to Open, encryption cipher to WEP, and the WEP key to “01.23.45.67.89”. Finally, the RADIUS server address, port and shared secret key is defined to tell the Access Point how to communicate with the Authentication Server.

There are usually a variety of these files. The primary file, named APConfig.xml, defines the standard configuration. Each of the other files define the Access Point configuration for a specific test suite. These files are named XXX\_APConfig.xml where the XXX serves to identify the test suite they are associated with.

### Test-Suite Configuration

#### Test-Suite Definitions

|  |
| --- |
| <TestSuite  TestSuiteName = "AuthMatrixTestSuite"  TestDllName = "WQMAuthMatrixTest.dll"  TestFacFuncName = "CreateAuthMatrixTestSuite"  TestPassCreteria = "100"  TestFailCreteria = "80"  TestSuiteWeight = "1.0"  NumberTestPasses = "1"  IsBadModes = "false"  IsBadKey = "false"  DisableAuthOpen = "false"  DisableAuthEAP = "false"  >  <Config>  <Command>  [WiFiTool](#_WiFiTool) –ejectxwifi  -adapter $(AdapterName) -power on  -list clear  </Command>  </Config> |

The first section in each test-suite file tells WiFi Metrics how to run the test suite (TestSuiteName, TestDllName, TestFacFuncName), defines tests-suite pass/fail criteria (TestPassCriteria, TestFailCriteria, TestSuiteWeight) and, finally, defines default test-case configuration parameters (NumberTestPasses, IsBadModes, IsBadKey).

In addition, many of the test-suites provide optional parameters for running the test in a special mode. In this excerpt, the DisableAuthEap option can be turned on to have the test automatically skip all the test cases requiring EAP (802.1X) authentication.

Finally, at the beginning of many of the test-suites is a <Config> section telling WiFi Metrics how to initialize the environment before running the test. In this example, the suite wants the command

[WiFiTool](#_WiFiTool) –ejectxwifi -adapter $(AdapterName) -power on -list clear

to be run. This will make sure the standard WiFi adapter is powered on, unload the “fake” WiFi adapter, and flush all the current WiFi Preferred Network configurations.

#### Test-Case Definitions

|  |
| --- |
| <TestCase  TuxTestIDOffset = "0"  TuxTestDescription = "Auth=Open Cipher=ClearText"  STAConfigAuth = "Open"  STAConfigEncr = "ClearText"  BvtTestCase = "true"  />  <TestCase  TuxTestIDOffset = "100"  TuxTestDescription = "Auth=Open Cipher=WEP 40-bit key (random)"  STAConfigAuth = "Open"  STAConfigEncr = "WEP"  STAConfigKeyIndex = "0"  STAConfigKey = "01.23.45.67.89"  BvtTestCase = "true"  /> |

These sections each describe a single test case. The examples define the first two tests in the Authentication Matrix test suite. Each test case starts with the test’s ID number offset and name. The test case ID numbers are calculated by adding the specified offset to the suite’s TstCaseIDStart from the [WiFiMetrics.xml](#_WiFiMetrics.xml) file. This allows the top-level file to control the overall test case IDs to guarantee that each case is assigned a unique number.

The rest of the parameters in each section define how the test case is to be initialized. The first example will test authenticating with Open authentication and no encryption. The second will test authenticating with Open authentication, WEP encryption and a WEP key of “01.23.45.67.89”.

Both of the examples are marked as BVT (Basic Viability) tests. This means they will be two of the relatively few cases run during BVT test runs.

Note that this technique allows the addition of new test cases or modification of existing cases by simply modifying this configuration file.

## Configuration Utilities

### APCTool – AP-Control Client

|  |
| --- |
| APCTool [-v] [-z] [-?] [other-commands]  General options:  -z Output test messages to the console.  -fl Output test messages to a file on the CE device.  -v Specifies the system to log with full verbosity.  -v[int] Specifies the system log with the given verbosity.  APControl options:  -lHost LAN server name/address (default "")  -lPort LAN server port (default "33331")  -wHost WiFi server name/address (default "")  -wPort WiFi server port (default "33331")  -wSSID WiFi server SSID (default “”)  -wAuth WiFi server authentication (default “Open”)  Open = Open authentication  Shared = Shared-key (WEP) authentication  WEP-802.1X = Open 802.1X authentication (Dynamic WEP)  WPA = WPA 802.1X authentication  WPA-PSK = WPA with Private Shared Key  WPA2 = WPA2 802.1X authentication  WPA2-PSK = WPA2 with Private Shared Key  -wEncr WiFi server encryption/cipher (default “ClearText”)  ClearText = no encryption  WEP = Wired Equivalent Privacy  TKIP = Temporal Key Interchange Protocol  AES = Advanced Encryption Standard  -wEapAuth EAP 802.1X Authentication mode (default “TLS”)  TLS = Transport Layer Security  MD5 = Message-Digest 5  PEAP = Protected Extensible Authentication Protocol  -wKey WiFi server encryption key (default “”)  for WEP, use hex form:  40-bit is a 10-digit hex number (ex: 12.34.56.78.90)  104-bit is a 26-digit hex number  (ex: 12.34.56.78.90.ab.cd.ef.12.34.56.78.90)  for TKIP or AES, use 8-63 char ASCII string (ex: asdfghjk)  -wKeyIndex WiFi server WEP key index (default “0”)  APCTool commands (default -q):  -q Queries current AP configurations  -i file Reads and sets AP config(s) from specified file  -o file Writes current AP configs to specified file |

This tool provides a command-line interface to the AP-Control Server. With no commands, the tool will contact the Server, query and display the current list of Access Points and RF-Attenuators being controlled.

The “-o” command is similar except after querying the current AP-Control Server configuration, the tool writes the information in XML format into the specified file.

Finally, the “-i” command reverses the process. It reads the configuration from the specified file and sends it to the AP-Control Server. The server will then perform whatever updates are necessary to update the Access Points and RF-Attenuators to match the new configurations.

This example is an excerpt from the NDIS One-Card Test Suite configuration file:

|  |
| --- |
| <Config>  <Filter  inputFile = "$(ScriptDirName)\NDT\_APConfig.xml"  outputFile = "$(TempDirName)\tempConfig.xml"  />  <Command>  APCTool -v -z  -i $(TempDirName)\tempConfig.xml  -lHost $(LANServerHost)  -lPort $(LANServerPort)  -wHost $(WiFiServerHost)  -wPort $(WiFiServerPort)  -wSSID $(WiFiServerSSID)  -wAuth $(WiFiServerAuth)  -wEncr $(WiFiServerCipher)  -wKey $(WiFiServerKey)  -wKeyIndex $(WiFiServerKeyIndex)  </Command>  </Config> |

The example first tells WiFi Metrics to filter the NDT\_APConfig.xml file in the process of copying it to tempConfig.xml. The filtering performs parameter substitution so strings like $(ConfigAP1Ssid) will be replaced with the proper values.

After the filtering, the example shows a command-line to have APCTool read the filtered AP configurations from tempConfig.xml, connect with the AP-Control Server and send the configuration updates.

When the update finishes, all the Access Points will be configured as required to run the NDIS One-Card Test.

### WiFiTool – WiFi Subsystem Configuration Utility

|  |
| --- |
| WiFiTool options:  -ejectxwifi Unloads the fake WiFi driver  -adapter    Adapter name (default: first WiFi  adapter found)  -power      on,off  -list       show,clear (Preferred list) |

WiFiTool is a simple utility for adjusting the basic state of the WiFi adapter(s) on a device. It is primarily used in WiFi Metrics to force the WiFi adapters into a known state before a test is launched.

An example of the usage comes from this section of the NDIS One-Card Test Suite configuration file:

|  |
| --- |
| <Config>  <Command>  WiFiTool -ejectxwifi -adapter $(AdapterName) -power on  -list clear  </Command>  </Config> |

The example tells WiFiTool to first unload the fake WiFi adapter then power on the normal adapter. Finally, it tells the command to clear the WZC Preferred Neworks list. (The last command is identical to [wzctool](#_WZCTool) –reset.)

## AP-Control Server Configuration

The AP-Control Server is an application running on Windows XP or Windows 2003 and attached to the [test network](#_Access_Points_and). On startup, the AP-Control Server registers itself, reads its configuration, and awaits requests to update the configurations of the Access Points and RF Attenuators under its control.

The following tables show the software requirements for the AP Control Server:

|  |  |
| --- | --- |
| ***Requirement*** | ***Description*** |
| apcontrol.exe | Runs the AP-Control Server |
| netall.dll | Provides networking and logging |
| wifiroamtest | This is the Server component of the [WiFiRoam Test](#_20301-20500:_WiFiRoam_–) |

The AP-Control Server is started by running the command:

**APControl [-z] [-v] [-s *server*] [-p *port*] [-k *regKey*]**

The following table shows the command line parameters for this executable.

|  |  |
| --- | --- |
| ***Command line parameter*** | ***Description*** |
| -k *regKey* | Specifies the registry key containing the initial configurations for the APs. The default value is **Software\Microsoft\CETest\APCTL**. |
| -p *port* | Specifies the TCP/IP port on which to connect. The default value is 33331. |
| -s *server* | Specifies the name or address of the server. The default value is **localhost**. |
| -z | Sends debug output to console. |
| -v | Specifies verbose debug output. |
| -? | Shows command line parameter information. |

The registry key specifies the initial configuration of the Access Point(s). Many settings can be specified here, but most of them are optional and, if supplied by the user, ignored. Instead, the server will retrieve them from the AP itself. The following table shows the required settings in the registry key.

| ***Required setting*** | ***Description*** |
| --- | --- |
| SSID | Contains the SSID of the AP. |
| Attenuator | Tells the server how to contact and control the RF attenuator. It has two components, the device-type and -address. The format is as follows:  *device\_type* **;** *device\_address*  Currently the *device\_type* should be one of **weinschel** or **manual**.  For **Weinschel** attenuators the *device\_address* is as follows:  *weinschel* **;** *com****C****-port****P***  C = serial port number (usually 1 or 2)  P = Weinschel device port number (1-4)  If you specify the type as **manual**, the attenuation driver displays a dialog box on the AP Control Server console that describes the new attenuation settings. You should manually update the AP's attenuator when this dialog appears. The tests allow approximately five minutes for an "OK" response before the test case fails and the test moves on. The format is as follows:  *manual* **;** *current* **:** *minimum* **:** *maximum*  current = current attenuation (in db)  minimum = minimum attenuation (in db)  maximum = maximum attenuation (in db) |
| Configurator | Tells the server how to contact the AP. It has two components, the basic AP type and its address. The format is as follows:  *device\_type* **;** *address* [**:** *user* **:** *password*]  Currently, the *device\_type* should be one of **cisco**, **dlink**, or **manual**.  The *address* is normally the IP address of the AP.  If the administrator name or password has been changed from the factory defaults, the new administrator name and password are specified immediately following the address.  If you specify the device type as **manual**, the AP driver displays a dialog box on the AP-Control Server console that describes the new AP settings. You should manually update the AP's configuration when this dialog appears. The test allows approximately five minutes for an "OK" response before the test case fails and the test moves on. |

Note that only the configurable Access Points (those within the RF Isolation enclosures) should be managed by the AP-Control Server. The AP(s) outside the enclosures should not be added to the registry.

The following is a sample registry file describing four access points:

|  |
| --- |
| Windows Registry Editor Version 5.00  [HKEY\_CURRENT\_USER\Software\Microsoft\CETest\APCTL]  [HKEY\_CURRENT\_USER\Software\Microsoft\CETest\APCTL\ROAM\_HOMEAP]  "Attenuator"="Weinschel;COM1-CHAN1"  "Configurator"="cisco;10.10.0.120:admin:Cisco"  "SSID"="ROAM\_HOMEAP"  [HKEY\_CURRENT\_USER\Software\Microsoft\CETest\APCTL\ROAM\_HOTSPOTAP]  "Attenuator"="Weinschel;COM1-CHAN2"  "Configurator"="dlink;10.10.0.140:admin:admin"  "SSID"="ROAM\_HOTSPOTAP"  [HKEY\_CURRENT\_USER\Software\Microsoft\CETest\APCTL\ROAM\_OFFICEAP1]  "Attenuator"="Weinschel;COM1-CHAN3"  "Configurator"="dlink;10.10.0.160:admin:admin"  "SSID"="ROAM\_OFFICEAP"  [HKEY\_CURRENT\_USER\Software\Microsoft\CETest\APCTL\ROAM\_OFFICEAP2]  "Attenuator"="Weinschel;COM1-CHAN4"  "Configurator"="dlink;10.10.0.180:admin:admin"  "SSID"="ROAM\_OFFICEAP" |

These registry settings describe four Access Points:

* **ROAM\_HOMEAP** is a Cisco AP with SSID "ROAM\_HOMEAP" which can be contacted at IP address 10.10.0.120 with the administrator name and password of “admin” and “Cisco”, respectively. The RF attenuation for the AP is controlled by a Weinschel attenuator connected to COM1 and the AP’s signals are routed through attenuator port 1.
* **ROAM\_HOTSPOTAP** is a D-Link AP with SSID "ROAM\_HOTSPOTAP" which can be contacted at IP address 10.10.0.140 with the administrator name and password of “admin” and “admin”, respectively. The RF attenuation for the AP is controlled by a Weinschel attenuator connected to COM1 and the AP’s signals are routed through attenuator port 2.
* **ROAM\_OFFICEAP1** is a D-Link AP with SSID "ROAM\_OFFICEAP" which can be contacted at IP address 10.10.0.160 with the administrator name and password of “admin” and “admin”, respectively. The RF attenuation for the AP is controlled by a Weinschel attenuator connected to COM1 and the AP’s signals are routed through attenuator port 3.
* **ROAM\_OFFICEAP2** is a D-Link AP with SSID "ROAM\_OFFICEAP" which can be contacted at IP address 10.10.0.180 with the administrator name and password of “admin” and “admin”, respectively. The RF attenuation for the AP is controlled by a Weinschel attenuator connected to COM1 and the AP’s signals are routed through attenuator port 4.

## Authentication Server Configuration

The Authentication (Radius) Server keeps a list of clients (Access Points). This list must include each Access Point used. Select and enter a passphrase for each AP; configure these keys into the corresponding APs so they can authenticate themselves with the Radius server during EAPOL key processing. The Authentication server (i.e., the Active Directory server) must have two special user accounts, one for TLS authentication and one for PEAP. By default, the tests use the following credentials for these accounts:

**TLS authentication**

* User name = **eaptls**
* Password = **eaptls**
* Domain = **wince**

**PEAP authentication**

* User name = **eappeap**
* Password = **eappeap**
* Domain = **wince**

For more detailed information see the [Authentication Server Setup](#_Appendix_D:_Authentication) appendix.

## NDT Server Configuration

For the [NDIS Two-Card Tests](#_Communication_Tests), we set up a second Support Device running the NDT Server application. With your Support Device, Connect to the WIFI\_OPEN fixed access point and obtain an IP address for the **NDTServerHost** parameter (in the [HdweSetup.xml](#_Communication_Parameters) file). Also, determine the Support Device’s WiFi adapter name and add this as the **NDTServerAdapter**.

Note that adding a reservation to your DHCP Server for the Support Device can simplify future test passes by giving the card the same IP address each time it connects. In the test default configuration files, we reserve IP address **10.10.0.116** for our NDT Server support NIC, **NDTServerHost**, and **ISPL21** for the WiFi adapter name, **NDTServerAdapter**.

Once the CE Device/CEPC is connected to the Test Environment, Start the Server with the following command:

**NDTServer –v Log\_Level:15**

The Log level default is 10 with a range of 0-15. In this case, we would like to see as much feedback from the server as possible so we use 15 as a log level.

|  |  |
| --- | --- |
| ***Command line parameter*** | ***Description*** |
| -s *server* | Specifies the name or address of the server. Default is **localhost**. |
| -v *Log\_Level:XX* | Specifies Log File verbosity (0-15, Default 10) |

Once the server is started, you will get a Notification box on the Mobile Device screen:

RELFSD: Opening file ndtserver.exe from desktop

NDTServer 054a0b26: Msg: ----------------------------------------------------------

NDTServer 054a0b26: Msg: NDTServer - Version 0.01

NDTServer 054a0b26: Msg: ----------------------------------------------------------

## NDIS Perf Server Configuration

The [NDIS Performance Test](#_1301-1350:_Ndis_perf_–) communicates with an NDIS Performance Server running on a desktop support system attached to the Test Network. (This usually is the system running the AP-Control Server.) You must install the MS\_NDP protocol driver on the supporting computer. You must also determine the bind name of the network interface for the supporting desktop computer.

#### To install the MS\_NDP protocol driver on the supporting desktop computer

1. Create a directory, and then copy to that directory the following files from <*Platform Builder installation path*>\Cepb\Wcetk\Ddtk\Desktop:

* Perf\_ndis.dll
* Ndp.sys
* Ndp.pdb
* Ndp.inf
* Snetcfg.exe
* Msvcr71.dll

1. To the directory that you created, copy the following files from <*Platform Builder installation path*>\Cepb\Wcetk\Ddtk\Desktop\Serial:

* Tux.exe
* Kato.dll

1. In the directory that you created, run the following command:

**snetcfg -l .\ndp.inf -c p -i ms\_ndp**

1. Run the following command to verify that the MS\_NDP protocol driver is installed:

**snetcfg -v -q ms\_ndp**

#### To uninstall the MS\_NDP protocol driver

1. In the directory that you created that contains the **Snetcfg.exe** file, run the following command:

**snetcfg -u ms\_ndp**

### Finding Names for a Network Interface

If the network interface for the tested Windows Embedded CE–based device is a wireless network interface and behaves as an access point (AP), the network interface for the supporting desktop computer must be a wireless network interface. Configure the wireless network interface for the supporting desktop computer to start in ad hoc network mode.

If the network interface for the tested Windows Embedded CE–based device acts as an AP, associate the network interface for the supporting desktop computer with the AP. Verify that no other wireless client connects to the AP.

#### To find the display name of the network interface for the supporting desktop computer

1. From the Start menu, choose **Control Panel**.
2. If Control Panel is set to Category View, choose **Network and Internet Connections**.
3. Choose **Network Connections**.
4. From the **View** menu, choose **Details**.
5. For the network connection, note the value under **Device Name**.

#### To find the bind name of the network interface for the supporting desktop computer

1. From a command prompt, navigate to the directory that you created that contains the files for the NDIS Performance Test.
2. Run the following command:

**tux -o -d perf\_ndis -c "-enum"**

1. From the list of network interfaces, note the bind name of the network interface whose display name matches the device name displayed in **Network Connections** of Control Panel. The bind name has the format **\Device\{***GUID***}**.

## Echo Service Configuration

The Echo Service runs on the same system as the AP-Control Server. It is used by various tests to send and receive UDP/IP and TCP/IP test traffic via the WiFi network. It will be contacted at the address defined in parameter [WiFiServerHost](#_Communication_Parameters).

The Echo Service allows tests to send data and receive back that same data to the originating device. The [Echo Protocol (RFC 862)](http://www.ietf.org/rfc/rfc0862.txt?number=862) specification is defined as:

TCP Based Echo Service:

One echo service is defined as a connection based application on TCP. A server listens for TCP connections on TCP port 7. Once a connection is established any data received is sent back. This continues until the calling user terminates the connection.

UDP Based Echo Service:

Another echo service is defined as a datagram based application on UDP. A server listens for UDP datagrams on UDP port 7. When a datagram is received, the data from it is sent back in an answering datagram.

To Install the Simple TCP/IP services, go to Add or Remove Programs under Windows Control Panel. Select Add Windows components and then the Networking Services Details button:



Create New firewall Exceptions for Echo service on Port 7 for both TCP and UDP. For Instructions on setting up Windows XP Firewall Exceptions see [Microsoft Support Article 842242](http://support.microsoft.com/kb/842242).

## Ad Hoc Station Configuration

N/A

## Prerequisites for Running the WiFi Metrics Test

The primary WiFi Metrics test is a Tux DLL which loads and runs lower-level DLLs and executables as instructed by a series of [configuration files](#_Configuration_Files). In general, running the test requires loading the executables and DLLs onto the device under test, modifying the DUT configuration file (see [DUTSetup.xml](#_DUTSetup.xml)) and running the top-level test.

The following tables show the software requirements for the WiFi Metrics Test:

| ***Requirement*** | ***Description*** |
| --- | --- |
| [apctool.exe](#_APCTool) | Command-line client for the AP-Control Server |
| enroll.exe | Retrieves authentication certificates from Auth Server |
| expbackofftest.dll | Exponential-Back-Off Test |
| kato.dll | Logging engine, required to log data |
| ndp.dll | NDIS Performance server |
| ndt.dll | NDIS Test Driver |
| ndt\_1c.dll | NDIS One-Card Test |
| ndt\_1c\_wlan.dll | NDIS One-Card WLAN Test |
| ndt\_2c.dll | NDIS Two-Card Test |
| ndt\_2c\_wlan.dll | NDIS Two-Card WLAN Test |
| netall.dll | Networking and logging utilities |
| perf\_ndis.dll | NDIS Performance Test |
| tooltalk.dll | Provides communication between test elements |
| tux.exe | Test harness, required to run test |
| wifimetrics.dll | Primary WiFi Metrics Test library |
| wifipower.dll | WiFi Power-Consumption Test |
| [wifitool.exe](#_WiFiTool) | WiFi Adapter Configuration Utility |
| wqmauthmatrixtest.dll | Authentication Matrix Test |
| wqmeapsimtest.dll | EAP-SIM Test |
| wqmlatencytest.dll | Packet Latency/Jitter Test |
| wqmssidtest.dll | Random and Broadcast SSID Test |
| wqmqostest.dll | QoS (Quality of Service) Test |
| wqmwifiroamtest.dll | Roaming and Stress Test |
| wqmwzctest.dll | Wireless Zero Config Test |
| [wzctool.exe](#_WZCTool) | Wireless Zero Config command-line utility |
| xwifi11b.dll | Fake-WiFi driver |
| [apconfig.xml](#_APConfig.xml) | Basic AP-Control AP configuration file |
| authmatrix\_test.xml | Authentication Matrix Test configuration file |
| [dutsetup.xml](#_DUTSetup.xml) | Device-Under-Test configuration file |
| eapsim\_test.xml | EAP-SIM Test configuration file |
| exp\_test.xml | Exponential Back-Off Test configuration file |
| [hdwesetup.xml](#_HdweSetup.xml) | Test Harness configuration file |
| ndis\_perf\_test.xml | NDIS Performance Test configuration file |
| ndt\_1c\_test.xml | NDIS One-Card Test configuration file |
| ndt\_1cwlan\_test.xml | NDIS One-Card WLAN configuration file |
| ndt\_2c\_test.xml | NDIS Two-Card Test configuration file |
| ndt\_2cwlan\_test.xml | NDIS Two-Card WLAN Test configuration file |
| ndt\_apconfig.xml | NDIS Test AP-Control AP configuration file |
| qos\_test.xml | QoS and Latency/Jitter Test configuration file |
| ssid\_test.xml | Random and Broadcast SSID Test configuration file |
| [wifimetrics.xml](#_WiFiMetrics.xml) | Primary WiFi Metrics Test configuration file |
| wifipower\_test.xml | WiFi Power-Consumption Test configuration file |
| wifiroam\_test.xml | Roaming and Stress Test configuration file |
| wzctest\_test.xml | Wireless Zero Config Test configuration file |

**Note:** When you run the WiFi Metrics Tests with Windows Mobile 6 Test Kit (WMTK) the Kit temporarily copies these files to the root directory of the target device. While the test runs, the test dynamically consumes program memory on the target device. Before running the test, verify that there is at least 1.0 megabytes (MB) of free storage memory on the target device. Also verify that there is at least 3.5 MB of free program memory on the target device. If there is not sufficient space in the root directory of the target device or there is not sufficient program memory, the test cannot run.

## Command-Line Parameters for the WiFi Metrics Test

The WiFiMetrics test executes the following command:

**tux –o –f \windows\wifimetrics.log –d wifimetrics.xml**

**–c “wifimetrics.xml [–v [suite:]name=value]”**

The following table shows the command line parameters for this executable.

| ***Command line parameter*** | ***Description*** |
| --- | --- |
| -o | Tells Tux to log the test output to the debug log (if any) |
| -f [wifimetrics.log](#_Metrics_Reporting_and) | Tells Tux to log a copy of the test output into the specified file |
| -d wifimetrics.dll | Tells Tux to load the wifimetrics.dll test library |
| -c [wifimetrics.xml](#_WiFiMetrics.xml) | Tells the WiFiMetrics test to load its primary configuration file from the specified file |
| -v name=value | Overrides the specified value in the XML configuration file(s). An optional “suite:” prefix indicates which test-suite is to be overridden. See below for more information about using these command line parameters. |

As specified in the [Configuration Files](#_Configuration_Files) section, the entire test configuration is described in XML files. The general mechanism for modifying this information is to modify those files. In certain circumstances, however, it is preferable to override the parameters by running the test with a special command line. The –v option has been provided for that purpose.

The syntax of the –v argument is either

**-v *name*=*value***

or

**-v *suite*:*name*=*value***

There can be no spaces in the *suite*, *name*, *value* or around the equals symbol.

The components of the command line argument are:

**suite:** This optional component limits the effect of this command line parameter to the specified test suite. For example, specifying

**–v AuthMatrix:DisableAuthEap=true**

will override the [DisableAuthEap](#_Parameters) parameter in the Authentication Matrix Test Suite from the default “false” to “true”. This will have no effect on another test-suite. The test suite names are case-insensitive – AuthMatrix and authmatrix are the same.

**name:** This component specifies the name of the parameter to be overridden. See [WiFi Metrics Tests](#_WiFi_Metrics_Tests) for a complete list of the configuration parameters. The parameter names are case-sensitive – DisableAuthEap, disableAuthEap and DisableAuthEAP are ***not*** the same.

**value:** This component specifies the new parameter value. Note that there can be no spaces in the value.

# Metrics Reporting and Storage

Two report files are produced by WiFi Metrics: a detailed test log and a result summary file. The test log is a normal Tux output log suitable for use by the standard WTT and CETK parsers. As such there is no need for a description here.

The summary file is encoded in XML:

|  |
| --- |
| <?xml version="1.0" standalone="yes"?>  <TestRun>  <Date> 01-24-2008 </Date>  <Time> 12:06:46 </Time>  <Device> ZippyPhone </Device>  <PassCriteria> 90 </PassCriteria>  <FailCriteria> 65 </FailCriteria>  <PassPercent> 85 </PassPercent>  <TestResult> MIXED </TestResult>  <TestGroup name = ”Stability”>  <PassCriteria> 90 </PassCriteria>  <FailCriteria> 65 </FailCriteria>  <TestWeight> 1.0 </TestWeight>  <PassPercent> 80 </PassPercent>  <TestResult> MIXED </TestResult>  <TotalPass> 20 </TotalPass>  <TotalSkip> 3 </TotalSkip>  <TotalFail> 5 </TotalFail>  <TestSuite name = ”Ndt\_1card”>  <PassCriteria> 90 </PassCriteria>  <FailCriteria> 65 </FailCriteria>  <TestWeight> 1.5 </TestWeight>  <PassPercent> 90 </PassPercent>  <TestResult> PASSED </TestResult>  <TotalPass> 9 </TotalPass>  <TotalSkip> 1 </TotalSkip>  <TotalFail> 1 </TotalFail>  <TestCase ID = ”1000” name = ”TestOpenClose”>  <TestResult> PASSED </TestResult>  </TestCase>  <TestCase ID = ”1001” name = ”TestSend”>  <TestResult> FAILED </TestResult>  </TestCase>  **...** |

By using XML, the test results can be transformed into any output format desired. In particular, the file can be directly loaded into a database or, with the addition of a simple XML schema file, added to a web site.

# WiFi Metrics Tests

## Common Test Parameters

Certain parameters are shared by all the test suites. These are defined in [WiFiMetrics.xml](#_WiFiMetrics.xml), [DUTSetup.xml](#_DUTSetup.xml), or [HdweSetup.xml](#_HdweSetup.xml).

### Directory Organization

Configuration File: **WiFiMetrics.xml**

|  |  |
| --- | --- |
| ***Parameter*** | ***Description*** |
| RootDirName | This is the name of the directory where all the test binaries are installed.  The default value is “\windows”. |
| ScriptDirName | This is the name of the directory where all the configuration files are installed.  The default value is “\release\WQMConfigs”. |
| TempDirName | This is the name of the directory where temporary files should be created.  The default value is “\windows”. |

### Default Run-Mode Parameters

Configuration File: **HdweSetup.xml**

| ***Parameter*** | ***Description*** |
| --- | --- |
| TestPassCriteria | This is the test-case pass-percentage required to receive a PASSED rating.  The default value is 90%. I.e., the test will pass if 90% or more of the test-cases pass. |
| TestFailCriteria | This is the test-case pass-percentage required to receive a FAILED rating.  The default value is 65%. I.e., the test will fail if 65% or fewer of the test-cases pass. |
| MaxFailCount | This is the number of consecutive test-case failures which will tell WiFi Metrics to abort the remaining cases and fail the overall test.  The default value is 10. |
| BvtTestRun | This flag indicates the test is being run in Basic Viability Test mode. In this mode, only test-cases marked with the **BvtTestCase** flag will be run.  The default value is false. |

### Device-Under-Test Configuration

Configuration File: **DUTSetup.xml**

|  |  |
| --- | --- |
| ***Parameter*** | ***Description*** |
| AdapterName | This is the name of the WiFi adapter.  The default value is “TNETWLN1”. |
| FakeAdapterName | This is the name of the fake-WiFi pseudo-driver.  The default value is “XWIFI11B1”. |
| ConnectionTimeLimit | This is the time, in seconds, the tests will wait for WiFi connections to be established.  The default value is 300 seconds. |
| NDTAdapterUnbind | This is used by the NDIS tests to disable unbinding of other protocol drivers from the test adapter before the test is run.  The default value is “-nounbind”. |

### Communication Parameters

Configuration File: **HdweSetup.xml**

| ***Parameter*** | ***Description*** |
| --- | --- |
| NDTServerHost | This is the IP address of the system running the NDTServer process. This address will be used by the [NDIS Two-Card Test](#_Ndt_2card_–_NDIS) to establish a connection to the Server.  The default value is “10.10.0.116”. |
| NDTServerAdapter | This is the name of the “support” adapter on the system running the NDTServer process. This will be used by the NDIS Two-Card Test to tell the Server which adapter to use for running the test.  The default value is “ISLP21”. |
| LANServerHost | This is the IP address or host name of the AP-Control Server. Note that this can only be used if the device-under-test has a LAN connection to the corresponding network.  This parameter and **WiFiServerSSID** are mutually exclusive. Either the device should connect to the AP-Control Server using the LAN (this parameter) or it should use the Fixed Access Point (WiFiServerSSID).  The default value is blank (not defined). |
| LANServerPort | This is the port number of the AP-Control Server running on the system at the LANServerHost address.  The default value is 33331. |
| WiFiServerHost | This is the IP address of the AP-Control Server on the WiFi Test Network. This is the address used by the tests to contact the Server and Echo Service after establishing a WiFi connection.  Note that if the device has a LAN connection this address should be on a different sub-net than the **LANServerHost**. Putting the device’s LAN and WiFi adapters on the same sub-net sets up the possibility that test packets which are supposed to travel over the WiFi network will, instead, be routed through the LAN.  The default value is 10.10.0.10. |
| WiFiServerPort | This is the port number of the AP-Control Server running on the system at the WiFiServerHost address.  The default value is 33331. |
| WiFiServerSSID | This is the SSID name of the Fixed Access Point. This will be used to establish a connection to the AP-Control Server through the WiFi Test Network.  This parameter and **LANServerHost** are mutually exclusive. Either the device should connect to the AP-Control Server using the Fixed Access Point (this parameter) or it should use the LAN adapter (LANServerHost).  The default value is the same as **FixedAPSSID**. |
| WiFiServerAuth | This is the authentication method used for associating with the Fixed Access Point. Options are **Open**, **Shared**, **WPA-PSK** and **WPA2-PSK**.  The default value is the same as **FixedAPAuth**. |
| WiFiServerEncr | This is the encryption method used for communicating with the Fixed Access Point. Options are **ClearText**, **WEP**, **TKIP** and **AES**.  The default value is the same as **FixedAPEncr**. |
| WiFiServerKey | This is the WEP key or TKIP/AES passphrase used for communicating with the Fixed Access Point.  For WEP keys, this should be either a 10-digit (40-bit) or 26-digit (104-bit) hexadecimal number. Examples:  10-digit: 12.34.56.78.90  26-digit: 12.34.56.78.90.12.34.56.78.90.ab.cd.ef  For TKIP or AES, this should be an ASCII string between 8 and 63 characters long. Examples:  8-characters: abcdefgh  20-characters: 12345678901234567890  The default value is the same as **FixedAPKey**. |
| WiFiServerKeyIndex | This is the WEP key index used for communicating with the Fixed Access Point. Options are 0, 1, 2, and 3.  The default value is the same as **FixedAPKeyIndex**. |

### Access Point Configuration

Configuration File: **HdweSetup.xml**

| ***Parameter*** | ***Description*** |
| --- | --- |
| SSIDSuffix | This suffix will be added to the tail of the standard SSID names. This is useful primarily when multiple WiFi Metrics test harnesses are running in close proximity to avoid AP-name collisions.  The default value is blank – no suffix will be added to the standard SSID names. |
| FixedAPSSID | This is the SSID name of the Fixed Access Point.  The default value is “WiFi\_OPEN”. |
| FixedAPAuth | This is the authentication method used for associating with the Fixed Access Point. Options are **Open**, **Shared**, **WPA-PSK** and **WPA2-PSK**.  The default value is Open. |
| FixedAPEncr | This is the encryption method used for communicating with the Fixed Access Point. Options are **ClearText**, **WEP**, **TKIP** and **AES**.  The default value is ClearText. |
| FixedAPKey | This is the WEP key or TKIP/AES passphrase used for communicating with the Fixed Access Point.  For WEP keys, this should be either a 10-digit (40-bit) or 26-digit (104-bit) hexadecimal number. Examples:  10-digit: 12.34.56.78.90  26-digit: 12.34.56.78.90.12.34.56.78.90.ab.cd.ef  For TKIP or AES, this should be an ASCII string between 8 and 63 characters long. Examples:  8-characters: abcdefgh  20-characters: 12345678901234567890  The default value is “01.23.45.67.89”. |
| FixedAPKeyIndex | This is the WEP key index used for communicating with the Fixed Access Point. Options are 0, 1, 2, and 3.  The default value is 0. |
| FixedAPWzcKey | This is the [WZCTool](#_WZCTool) equivalent of the FixedAPKeyIndex and FixedAPKey.  For WEP keys, this is the key-index follow by the hexadecimal key. See [WZCTool](#_WZCTool) for more information.  For TKIP and AES, this is the same as FixedAPKey.  The default value is “1/0x0123456789”. |
| ConfigAP1Name | This is the name of the first configurable Access Point in the AP-Control Server’s registry.  The default value is “ROAM\_HOMEAP”. |
| ConfigAP1SSID | This is the default SSID name of the first configurable Access Point.  The default value is “ROAM\_HOMEAP$(SSIDSuffix)”. |
| ConfigAP2Name | This is the name of the second configurable Access Point in the AP-Control Server’s registry.  The default value is “ROAM\_HOTSPOTAP”. |
| ConfigAP2SSID | This is the default SSID name of the second configurable Access Point.  The default value is “ROAM\_HOTSPOTAP$(SSIDSuffix)”. |
| ConfigAP3Name | This is the name of the third configurable Access Point in the AP-Control Server’s registry.  The default value is “ROAM\_OFFICEAP1”. |
| ConfigAP3SSID | This is the default SSID name of the third configurable Access Point.  The default value is “ROAM\_OFFICEAP$(SSIDSuffix)”. |
| ConfigAP4Name | This is the name of the fourth configurable Access Point in the AP-Control Server’s registry.  The default value is “ROAM\_OFFICEAP2”. |
| ConfigAP4SSID | This is the default SSID name of the fourth configurable Access Point.  The default value is “ROAM\_OFFICEAP$(SSIDSuffix)”. |
| APNamesList | Many of the tests choose from this comma-separated list of access point (AP) names when deciding which of the AP-Control Server’s APs to utilize for their testing.  Normally, the tests select from among all the APs controlled by the Server. This option forces the application to select from the listed APs. If there are multiple APs to choose from, the test suite normally decides which to use in the following order:   1. If there is an AP with a matching security mode, use it. 2. Otherwise, if not all the APs support the security modes, use the first that does. 3. Otherwise, use the first AP in the list. |

### Authentication Certificate Enrollment

Configuration File: **DUTSetup.xml and HdweSetup.xml**

| ***Parameter*** | ***Description*** |
| --- | --- |
| EnrollHostName | This is the IP address or host name of the Authentication Server.  The default value is “10.10.0.1”. |
| EnrollHostPort | This is the port number for the RADIUS / IAS service on the Authentication Server.  The default value is “1812”. |
| EnrollPassword | This is the passphrase used by the Access Points to authenticate themselves with the RADIUS server.  The default value is “0123456789”. |
| EnrollCommand | This is the name of the certificate-enrollment command.  The default value is “enroll.exe”. |
| EnrollRootDlg | This is the title of the “set root certificate” dialog box.  The default value is “Root Certificate Store”. |
| EnrollToolDlg | This is the title of the “insert new certificate” dialog box.  The default value is “Enrollment Tool”. |
| EnrollTime | This is the time, in milliseconds, to wait for the enroll command to finish.  The default value is 180000ms (3 minutes). |

### EAP Authentication / Login Credentials

Configuration File: **DUTSetup.xml and HdweSetup.xml**

| ***Parameter*** | ***Description*** |
| --- | --- |
| PEAPAccountCreds | These are the credentials for authenticating the test user with the RADIUS server when using PEAP authentication. The format is *user\_name*:*password*:*domain*.  The default value is “eappeap:eappeap:wince”. |
| TLSAccountCreds | These are the credentials for authenticating the test user with the RADIUS server when using TLS authentication. The format is *user\_name*:*password*:*domain*.  The default value is “eaptls:eaptls:wince”. |
| UserLogOnDlg | This is the title of the user login dialog box.  The default value is “User Logon”. |
| NewPassWdDlg | This is the title of the network password dialog box.  The default is “Enter Network Password”. |
| LogonCloseTime | This is the maximum time, in milliseconds, to wait for the user-logon dialog box to close after we enter our credentials.  The default value is 300000ms (5 minutes). |

## Stability Tests

### 1001-1050: Ndt\_1card – NDIS One-Card Test

Configuration File: **ndt\_1c\_test.xml**

#### Overview

The One-Card Network Card Miniport Driver Test assesses the functionality of a miniport driver for a single network card. You can also use this test to verify that the driver supports Network Driver Interface Specification (NDIS) functionality.

This test is comprised of two binaries, **Ndt.dll** and **Ndt\_1c.dll**. The **Ndt.dll** binary file is a protocol driver that binds to the test and support cards. The protocol driver communicates with the underlying miniport drivers through an NDIS wrapper and registers as a stream driver. The **Ndt\_1c.dll** binary file controls the test itself.

#### Parameters

| ***Parameter*** | ***Description*** |
| --- | --- |
| NDTLogPackets=–packets | Logs information when a test confirms that a packet has been sent or received.  The default value is blank – test does not log packets. |
| NDTPacketUnbind=–nounbind | Disables unbinding of other protocol drivers from the test adapter before the test is run.  The default value is “-nounbind”. (See [DUTSetup.xml](#_DUTSetup.xml).) |
| NDTCancelTest=–cancel | Instructs test case 1007 to fail when no packets are canceled.  The default value is blank - test case 1007 does not fail when no packets are canceled. |
| NDTFaultTest=–fault | Includes test case 1008, which is skipped by default.  Only ISA and PCI network cards support test case 1008.  The default value is blank - test case 1008 is skipped. |

#### Test Cases

| ***Test Case*** | ***Description*** |
| --- | --- |
| 1001 | Open\Close  Tests the ability to open and close an adapter multiple times. A Miniport driver is shielded from the opening of an adapter by the Ndt.dll protocol driver. As a result, this test case tests the Ndt.dll protocol driver rather than the miniport driver. This test case opens and closes one instance of **NdisOpen** 16 times, and then opens and closes 128 concurrent instances of **NdisOpen** 16 times. This test fails if the miniport driver has a problem with multiple open instances of **NdisOpen**. |
| 1002 | Send  Tests the ability to send data both singularly and in bursts. The test case attempts to send data to the various address types supported by the media type of the current driver. This test case fails if a problem occurs with sending packets. |
| 1003 | Loopback send  Tests for the ability to receive loopback packets with a variety of address types on multiple filter settings. The test uses one open instance to send loopback packets and eight instances to receive the packets. Each of the eight instances receiving the packets has a different filter setting, which allows for all supported filter settings to be tested quickly. This test case also verifies that no open instance receives a packet that it should not be receiving. This test case fails if a problem occurs with a filter setting in a driver. |
| 1004 | Loopback stress  Creates packets with various buffer configurations in order to perform 10 instances of the stress test on the loopback packets. This test case fails if a memory leak is detected. |
| 1005 | Set multicast  Tests the ability of the Ethernet and Fiber Distributed Data Interface (FDDI) drivers to create multiple multicast addresses. The test does not verify that the card is able to receive on each of the different addresses. The test verifies only that multicast addresses can be set and deleted. |
| 1006 | Reset  Attempts to reset the network card multiple times while simultaneously sending large numbers of packets. The test also verifies that the card can reset itself in order to properly handle packets that are ready to send when interrupted by a reset. This test case fails if the operation that resets the network card is implemented incorrectly. |
| 1007 | Cancel send  Runs the performance command with a flag that causes the Ndt.dll protocol driver to cancel packets. The performance command queues 100 packets to send. In the next packet to send, the performance command sets the cancel identification, and then attempts to cancel the send operation. This test case fails if an improper cancellation of a packet occurs. |
| 1008 | Fault handling  Sets bits in the registry for the network card driver using the fault injection NDIS technology.  These bits cause NDIS to fail the **NdisMAllocateMapRegisters**, **NdisMRegisterInterrupt**, **NdisMAllocateSharedMemory**, **NdisMMapIoSpace**, **NdisMRegisterIoPortRange**, **ReadNdisGetSetBusConfigSpace**, **WriteNdisGetSetBusConfigSpace**, and **NdisMInitializeScatterGatherDma** functions.  The driver should not load correctly unless it does not call a particular function. |
| 1009 | Object identifiers  Performs a series of **NdisRequest** function calls to the driver. Verifies that the driver supports the querying of all required object identifiers. |
| 1010 | 64 bit object identifiers  Tests the OID\_GEN\_XMIT\_OK and OID\_GEN\_RCV\_OK object identifiers to verify that all queries are handled properly. Each object identifier is queried three times. The object identifier is queried once with a null buffer, once with a 4-byte buffer and once with an 8-byte buffer. |
| 1011 | Suspend and then resume  Tests the behavior of the network card driver when the test suspends and then resumes the operating system (OS). If the run-time image does not support the IOCTL\_HAL\_ENABLE\_WAKE IOCTL or does not wake in response to a SYSINTR\_RTC\_ALARM interrupt, this test case is skipped. This test case suspends and then resumes the OS 5 times and then attempts to send data over the network card driver to verify that the driver is functional. |
| 1012 | Stress suspend and then resume  Tests power management of the network card driver and stresses the network card driver under suspend and resume operations.  This test case has three threads. The main thread performs suspend and resume operations. The second thread sends data continuously over the network card driver. The third thread queries object identifiers (OIDs) continuously.  If the network card driver supports power management, the main thread requests that Device Manager put the network card into a D4 state. After one second, the main thread requests that Device Manager restore the network card to a D0 state. If the network card driver does not support power management, the main thread performs a reset operation on the network card and then waits for a random interval of time.  This test case assesses the ability of the network card driver to process send requests and OID requests while undergoing power transition. This test case performs 25 suspend and then resume operations. |
| 1013 | Reset on resume  Tests the ability of the network card driver to restore its original settings when a resume operation resets the network card driver. This test case first sets a packet filter, multicast list size, and look-ahead buffer size. This test case then adds multicast addresses to the multicast list. After a reset operation, this test case verifies that the network card driver preserves its original settings. |
| 101 | Ndt\_1card\_Sleep\_Block |

## Communication Tests

### 1101-1150: Ndt\_2card – NDIS Two-Card Test

Configuration File: **ndt\_2c\_test.xml**

#### Overview:

The Two-Card Network Card Miniport Driver Test assesses the functionality of a miniport driver on a target device with two network cards.

#### Parameters

| ***Parameter*** | ***Description*** |
| --- | --- |
| NDTLogPackets=–packets | Log information when a test confirms that a packet has been sent or received.  The default value is blank – test does not log packets. |
| NDTPacketUnbind=–nounbind | Disable unbinding of other protocol drivers from the test adapter before the test is run.  The default value is “-nounbind”. (See [DUTSetup.xml](#_DUTSetup.xml).) |
| NDTBeatDelay=*delay* | Print debug output at intervals specified by the value of *delay*. The debug output contains information about the state of packet buffers for the test. The value of *delay* is given in milliseconds.  The default value is 0. |
| NDTSkipStress=–nostress | Skip test cases 1105 and 1106.  The default value is blank – test does not skip stress tests. |
| NDTPacketDelay=*interval* | Set the interval, in milliseconds (ms), between packets sent during test cases 1105 and 1106.  The default interval is 10 ms. |

#### Test Cases

| ***Test Case*** | ***Description*** |
| --- | --- |
| 1101 | Send packets  Sends packets using the **NdisSendPackets** function. The test uses various burst and packet sizes and logs a failure if any problems occur during **NdisSendPackets**. This test case uses either a minimum packet size of 64/96 bytes, the maximum packet size supported by the medium, or an average of these two sizes. |
| 1102 | Receive packets  Tests whether the card is able to correctly receive packets on its hardware Media Access Control (MAC) address. |
| 1103 | Filter receive  Tests whether the card is able to correctly receive packets with various addressing types. The test uses one open instance to send and eight instances to receive. Each of the eight receiving instances has a different filter setting, which allows all supported filter settings to be tested quickly and verifies that an open instance does not receive a packet that it should not receive. |
| 1104 | Multicast receive  Tests whether the card is able to receive on as many different multicast addresses as the card claims to support. The test uses all available multicast addresses and attempts to send packets to each of those addresses. The test also verifies that packets are only received on the multicast addresses that are active. |
| 1105 | Stress send  Executes a stress test between the test card and the support card on the target device. The test card sends packets and the support card receives packets. The test runs for 10 iterations with various buffering options. The test also verifies that the test card can send packets of differing size at a faster rate and can simultaneously receive different types of acknowledgement packets. Packet loss may occur in this test. |
| 1106 | Stress receive  Executes a stress test between the test card and the support card on the target device. The support card sends packets and the test card receives packets. The test runs for five iterations with various buffering options. The test also verifies that the test card can receive packets of differing size at a faster rate and can simultaneously send different types of acknowledgement packets. Packet loss is common during this test. The main criterion for success is that the miniport driver should be able to handle send and receive requests with various buffer configurations. |

## Security Tests

### 2000-15000: AuthMatrix – Authentication Matrix Test

Configuration File: **authMatrix.xml**

#### Overview

The Wi-Fi Authentication Matrix Test validates the proper operation of a Windows Mobile powered device by exercising all valid combinations of the authentication and encryption protocols in connection with an access point (AP), as well as ensuring that the device will not connect to the AP when the configuration is incorrect.

#### Parameters

| ***Parameter*** | ***Description*** |
| --- | --- |
| DisableAuthOpen=true | Disables (skips) **Open** System (no authentication) tests |
| DisableAuthShared=true | Disables **Shared** (WEP authentication) tests |
| DisableAuth802=true | Disables 802.1X (dynamic WEP) tests |
| DisableAuthEAP=true | Disables EAP (Radius) tests (**WEP 802.1X**, **WPA** and **WPA2**) |
| DisableAuthPSK=true | Disables **PSK** tests (**WPA-PSK** and **WPA2-PSK**) |
| DisableAuthWPA2=true | Disables **WPA2** tests (**AES**, **WPA2** and **WPA2-PSK**) |
| DisableCipherClear=true | Disables **ClearText** (no encryption) tests |
| DisableCipherWEP=true | Disables **WEP** encryption tests |
| DisableCipherTKIP=true | Disables **TKIP** encryption tests |
| DisableCipherAES=true | Disables **AES** encryption tests |
| DisableEapAuthPEAP=true | Disables **PEAP** EAP-authentication tests |
| DisableEapAuthTLS=true | Disables **TLS** EAP-authentication tests |

#### Test Cases

| ***Test Case*** | ***Description*** |
| --- | --- |
| 2000 | Auth=Open; Cipher=ClearText. |
| 2100 | Auth=Open; Cipher=WEP 40-bit key (random). |
| 2101 | Auth=Open Cipher=WEP 40-bit key (random) AP Auth=Shared(negative test) |
| 2110 | Auth=Open; Cipher=WEP 40-bit key (semi-null). |
| 2111 | Auth=Open Cipher=WEP 40-bit key (semi-null), AP Auth=Shared(negative test) |
| 2120 | Auth=Open; Cipher=WEP 40-bit key (semi-ones). |
| 2121 | Auth=Open Cipher=WEP 40-bit key (semi-ones), AP Auth=Shared(negative test) |
| 2130 | Auth=Open; Cipher=WEP 104-bit key (random) |
| 2131 | Auth=Open Cipher=WEP 104-bit key (random), AP Auth=Shared(negative test) |
| 2140 | Auth=Open; Cipher=WEP 104-bit key (semi-null) |
| 2141 | Auth=Open Cipher=WEP 104-bit key (semi-null), AP Auth=Shared(negative test) |
| 2150 | Auth=Open; Cipher=WEP 104-bit key (semi-ones) |
| 2151 | Auth=Open Cipher=WEP 104-bit key (semi-ones), AP Auth=Shared(negative test) |
| 2200 | (Test case should fail) AP: Auth=Open, Cipher=WEP; STA: Auth=Open, Cipher=TKIP. |
| 2210 | (Test case should fail) STA: Auth=Open Cipher=WEP AP: Auth=Open Cipher=WEP Invalid Key |
| 2220 | (Test case should fail) STA: Auth=Open Cipher=WEP AP: Auth=Shared Invalid Key |
| 2300 | (Test case should fail) AP: Auth=Open, Cipher=WEP; STA: Auth=Open, Cipher=AES. |
| 3000 | (Test case should fail) AP: Auth=Shared Cipher=WEP; STA: Auth=Shared, Cipher=ClearText. |
| 3100 | Auth=Shared; Cipher=WEP 40-bit key (random). |
| 3101 | Auth=Shared Cipher=WEP 40-bit key (random); AP Auth=Open |
| 3110 | Auth=Shared; Cipher=WEP 40-bit key (semi-null). |
| 3111 | Auth=Shared Cipher=WEP 40-bit key (semi-null), AP Auth=Open" |
| 3120 | Auth=Shared; Cipher=WEP 40-bit key (semi-ones). |
| 3121 | Auth=Shared Cipher=WEP 40-bit key (semi-ones), AP Auth=Open |
| 3130 | Auth=Shared; Cipher=WEP 104-bit key (random). |
| 3131 | Auth=Shared Cipher=WEP 104-bit key (random), AP Auth=Open |
| 3140 | Auth=Shared; Cipher=WEP 104-bit key (random). |
| 3141 | Auth=Shared Cipher=WEP 104-bit key (semi-null), AP Auth=Open |
| 3150 | Auth=Shared; Cipher=WEP 104-bit key (semi-ones). |
| 3151 | Auth=Shared Cipher=WEP 104-bit key (semi-ones), AP Auth=Open |
| 3200 | (Test case should fail) AP: Auth=Shared, Cipher=WEP; STA: Auth=Shared, Cipher=TKIP. |
| 3210 | (Negative test) STA/AP: Auth=Shared Cipher=WEP Shared Invalid Wep Key |
| 3220 | (Negative test) AP: Auth=Shared Cipher=WEP STA: Open Invalid Wep Key |
| 3300 | (Test case should fail) AP: Auth=Shared, Cipher=WEP; STA: Auth=Shared, Cipher=AES. |
| 5000 | (Test case should fail) AP: Auth=WPA, Cipher=TKIP; STA: Auth=WPA, Cipher=ClearText. |
| 5100 | (Test case should fail) AP: Auth=WPA, Cipher=TKIP; STA: Auth=WPA, Cipher=WEP. |
| 5220 | Auth=WPA; Cipher=TKIP EAP=PEAP. |
| 5230 | Auth=WPA Cipher=AES EAP=PEAP --- Eap Cred prefilled |
| 5330 | Auth=WPA Cipher=TKIP EAP=PEAP |
| 5340 | Auth=WPA Cipher=AES EAP=PEAP |
| 6000 | (Test case should fail) AP: Auth=WPA\_PSK, Cipher=TKIP; STA: Auth=WPA\_PSK, Cipher=ClearText. |
| 6100 | (Test case should fail) AP: Auth=WPA\_PSK, Cipher=TKIP; STA: Auth=WPA\_PSK, Cipher=WEP. |
| 6200 | Auth=WPA\_PSK; Cipher=TKIP 63-digit passphrase (random) |
| 6210 | Auth=WPA\_PSK; Cipher=TKIP 8-digit passphrase (random). |
| 6220 | (Test case should fail) Auth=WPA\_PSK; Cipher=TKIP 7-digit passphrase. |
| 6230 | Auth=WPA\_PSK; Cipher=TKIP 63-digit passphrase (semi-ones). |
| 6240 | Auth=WPA\_PSK; Cipher=TKIP 8-digit passphrase (semi-ones). |
| 6250 | (Test case should fail) Auth=WPA\_PSK; Cipher=TKIP 64-digit passphrase. |
| 6260 | Auth=WPA\_PSK; Cipher=TKIP 63-digit passphrase (semi-null). |
| 6270 | Auth=WPA\_PSK; Cipher=TKIP 8-digit passphrase (semi-null). |
| 6300 | Auth=WPA\_PSK; Cipher=AES 63-digit passphrase (random). |
| 6310 | Auth=WPA\_PSK; Cipher=AES 8-digit passphrase (random). |
| 6320 | (Test case should fail) Auth=WPA\_PSK; Cipher=AES 7-digit passphrase. |
| 6330 | Auth=WPA\_PSK; Cipher=AES 63-digit passphrase (semi-ones). |
| 6340 | Auth=WPA\_PSK; Cipher=AES 8-digit passphrase (semi-ones). |
| 6350 | (Test case should fail) Auth=WPA\_PSK; Cipher=AES 64-digit passphrase. |
| 6360 | Auth=WPA\_PSK; Cipher=AES 63-digit passphrase (semi-null). |
| 6370 | Auth=WPA\_PSK; Cipher=AES 8-digit passphrase (semi-null). |
| 7000 | (Test case should fail) AP: Auth=WPA2, Cipher=TKIP; STA: Auth=WPA2, Cipher=ClearText. |
| 7100 | (Test case should fail) AP: Auth=WPA2, Cipher=TKIP; STA: Auth=WPA2, Cipher=WEP. |
| 7220 | Auth=WPA2; Cipher=TKIP EAP=PEAP. |
| 7230 | Auth=WPA2 Cipher=AES EAP=PEAP |
| 7320 | Auth=WPA2; Cipher=AES EAP=PEAP. |
| 7330 | Auth=WPA2 Cipher=AES EAP=PEAP |
| 8000 | (Test case should fail) AP: Auth=WPA2\_PSK, Cipher=TKIP; STA: Auth=WPA2\_PSK, Cipher=ClearText. |
| 8100 | (Test case should fail) AP: Auth=WPA2\_PSK, Cipher=TKIP; STA: Auth=WPA2\_PSK, Cipher=ClearText. |
| 8200 | Auth=WPA2\_PSK; Cipher=TKIP 63-digit passphrase (random). |
| 8210 | Auth=WPA2\_PSK; Cipher=TKIP 8-digit passphrase (random). |
| 8220 | (Test case should fail) Auth=WPA2\_PSK; Cipher=TKIP 7-digit passphrase. |
| 8230 | Auth=WPA2\_PSK; Cipher=TKIP 63-digit passphrase (semi-ones). |
| 8240 | Auth=WPA2\_PSK; Cipher=TKIP 8-digit passphrase (semi-ones). |
| 8250 | (Test case should fail) Auth=WPA2\_PSK; Cipher=TKIP 64-digit passphrase. |
| 8260 | Auth=WPA2\_PSK; Cipher=TKIP 63-digit passphrase (semi-null). |
| 8270 | Auth=WPA2\_PSK; Cipher=TKIP 8-digit passphrase (semi-null). |
| 8300 | Auth=WPA2\_PSK; Cipher=AES 63-digit passphrase (random). |
| 8310 | Auth=WPA2\_PSK; Cipher=AES 8-digit passphrase (random). |
| 8320 | (Test case should fail) Auth=WPA2\_PSK; Cipher=AES 7-digit passphrase. |
| 8330 | Auth=WPA2\_PSK; Cipher=AES 63-digit passphrase (semi-ones). |
| 8340 | Auth=WPA2\_PSK; Cipher=AES 8-digit passphrase (semi-ones). |
| 8350 | (Test case should fail) Auth=WPA2\_PSK; Cipher=AES 64-digit passphrase. |
| 8360 | Auth=WPA2\_PSK; Cipher=AES 63-digit passphrase (semi-null). |
| 8370 | Auth=WPA2\_PSK; Cipher=AES 8-digit passphrase (semi-null). |
| 9000 | Authmatrix Stress Test |

# Appendix A: Supported Access Points

These Access Point types are currently supported by the AP-Control Server:

|  |  |  |
| --- | --- | --- |
| ***Vendor/Model*** | ***Firmware*** | ***Supported Modes*** |
| Cisco Aironet  AIR-1232AG | 12.3(8)-JEA1 | Open  Shared  WEP-802.1X  WPA  WPA-AES  WPA-PSK  WPA2  WPA2-PSK |
| Cisco Aironet  AIR-1240AG | 12.3(4)-JA or Later;  LWAPP3.1 or later. | Open  Shared  WEP-802.1X  WPA  WPA-AES  WPA-PSK  WPA2  WPA2-PSK |
| D-Link  DWL-3200 | Version 2.10 or 2.20 | Open  Shared  WPA  WPA-AES  WPA-PSK  WPA2  WPA2-PSK |

Four Access Points must be selected from this list. These APs will be re-configured throughout the test. These APs will also reside inside the [RF-Enclosures](#_Appendix_C:_Example) and their RF signal strength will be controlled by the [RF-Attenuator](#_Appendix_B:_Supported_1).

One or two additional Access Points are needed. These can be any type of Access Point. They do not have to be chosen from the Supported APs table:

**Fixed AP:** This AP will be used to contact the AP-Control Server, the NDIS Performance server and so on. It must be available and configured as follows at all times:

1. Set the SSID to **WiFi\_OPEN**
2. Set the Authentication to **Open**
3. Set the Encryption to **Disabled**
4. Set the DHCP server to **Disabled**

**NDIS Two-Card Test AP5:** This AP will only be used during the NDIS Two-Card Test ***if*** the WiFi adapter of the device being tested supports WPA2 authentication. If required, it must be configured as follows:

1. Set the SSID to **NDTEST\_WPA2\_AP1**
2. Set the WPA2-PSK status to **Enabled**
3. Set the Pre Shared key to **0123456789**
4. Set WPA Encryption to **AES**
5. Set the DHCP server to **Disabled**

# Appendix B: Supported RF Attenuators

These RF-Attenuators types are currently supported by the AP-Control Server:

|  |  |  |
| --- | --- | --- |
| ***Vendor/Model*** | ***Firmware*** | ***Description*** |
| Aeroflex Weinschel  8310-352-4-T | N/A | 6 GHz box, 4 channels,  0 to 103 dB in 1 dB steps per channel |

# Appendix C: Example Hardware Bill of Materials

## RF Attenuation



|  |  |  |  |
| --- | --- | --- | --- |
| ***Ref*** | ***Vendor/Model*** | ***Description*** | ***Quantity*** |
| 1 | Centurion  WTS2450-RPSMA | WTS WLAN antenna  Tri-band, 2.4 to 2.5 GHz and 4.9 to  5.875 GHz with reverse polarity  SMA male connector. | 4 |
| 2 | Amphenol  132 169RP | Connex adapter,  reverse polarity SMA female  SMA female adapter. | 4 |
| 3 | EZ402SMAP-48 | EZ Form Cable Assembly,  48 inches of EZ Flex 402 cable with  SMA male connectors on both ends | 4 |
| 4 | Radiall  R191 329 000 N | N male to SMA female adapter | 8 |
| 5 | Aeroflex Weinschel  8310-352-4-T | Programmable RF Attenuator  6 GHz box, 4 channels,  0 to 103 dB in 1 dB steps per channel | 1 |
| 6 | EZ402SMAP-48 | EZ Form Cable Assembly,  48 inches of EZ Flex 402 cable with  SMA male connectors on both ends | 4 |

## RF Isolation Enclosures (2)



| ***Ref*** | ***Vendor/Model*** | ***Description*** | ***Quantity*** |
| --- | --- | --- | --- |
| 1 | Ramsey Electronics  STE3300M  MISC55  CONN157  CONN207  STERJ45  STERF13  PS18 | RF Enclosure  STE Plate Mod.  SMA Jack to Jack RoHS  DB9 100 PF Pi Filtered Connector  RJ45-DB9 Interface (input/output)  4 Terminal Assembly 100VDC  Power Strip | 2 |
| 2 | Amphenol  132 171 RP-10 | Connex SMA female to SMA  reverse polarity male adapter | 8 |
| 3 | EZ402SMAP-6 | EZ Form Cable Assembly,  6 inches of EZ Flex 402 cable with  SMA male connectors on both ends | 8 |
| 4 | XPD10-2-8-2S | XMA 2:1 Power Divider/Combiner,  2 to 8 GHz, SMA female connectors | 4 |
| 5 | EZ402SMAP-12 | EZ Form Cable Assembly,  12 inches of EZ Flex 402 cable with  SMA male connectors on both ends | 4 |
| 6 | Misc | 4-port mini network hub | 2 |
| 7 | Misc | 12-inch RJ45 network cable | 2 |

Item 2 adapts the DLink DWL 3200 Access Points antenna connectors to fit the SMA male connectors on the RF cables (item 3). Wiring in a different type of Access Point will likely require different adapters.

## Access Points and Test Network



| ***Ref*** | ***Vendor/Model*** | ***Description*** | ***Quantity*** |
| --- | --- | --- | --- |
| 1 | Misc | Generic Access Point  Second unit must support WPA2 | 2 |
| 2 | Cisco Aironet  AIR-1232AG | ABG Access Point  (see [Supported Access Points](#_Appendix_A:_Supported)) | 1 |
| 3 | DLink  DWL 3200 | BG Access Point  (see [Supported Access Points](#_Appendix_A:_Supported)) | 3 |
| 4 | Misc | 8-port Network Hub | 1 |
| 5 | Misc | Windows 2003 Server, running  Active Directory, DHCP Server, and  Internet Authentication Service (RADIUS) | 1 |
| 6 | Misc | Windows XP running  Windows Visual Studio,  Windows Mobile Platform Builder,  Simple TCP/IP Services (Echo Server), and  Access Point Control Server | 1 |

The test network should be isolated from corporate or other networks to avoid interference with the tests and to isolate the test network to avoid exposing sensitive information over the unsecured Access Points.

# Appendix D: Authentication Server Setup

## Overview

This document covers a start to finish, step by step guide to setting up an Authentication Server, also Known and a RADIUS or AAA server, for use with the Windows Mobile test kits.   
This will include setting up the following servers and services.

**Active Directory**

**DNS Server**

**DHCP Server**

**Internet Information Service (IIS)**

**Certificate Services**

**Internet Authentication Service (IAS)**

## Pre-requisites

Windows Server 2003, Standard or Enterprise, must be installed and fully updated.

Once the Windows 2003 Server is installed and has been logged onto for the first time, you will see a ‘Manage Your server’ Window. For Simplicity Purposes, this document will work off this administration screen.



Click on the ‘Add or remove a role’ Icon to add services to your server.



Read the prerequisites and click ‘Next’

## Install and Configure Active Directory and DNS Server



First thing that needs to be done is to promote the server to a domain controller. On Windows 2003 server, this will also install and configure Active Directory.

  
Click the ‘Next’ button.



This will be on a private, isolated network and therefore needs to be a root domain controller for a new forest.



Choose ‘Domain in a new Forest’



Since this is a test environment, use the .local label for the full DNS name. It’s a more secure configuration because the .local label is not registered for use on the Internet and there is no need or attempt to authenticate to a higher domain.



Database and Log Folder locations can generally be left as defaults. If you have a need for them to be on another local drive or folder you can change the path. Note: if you change the drive of the database, make sure it is a local resource and not a network mapped path.

  
In our test environment, setting a shared system volume is irrelevant as there will not be any additional domain controllers. Click ‘Next’

  
Windows Server 2003 will automatically configure DNS to work with Active Directory if it is installed during the AD install otherwise it will need to be installed and manually configured later. Note: it is recommended that DNS is installed at this point.



Windows Mobile test environment only supports the use of Windows 2000 and newer so choosing Permissions compatible only with Windows 2000 or 2003 is the correct choice.

  
In a test environment this will generally not be used. For our purposes, we will set it the same as the machine administrator account that was used during the Windows Server 2003 Installation. (Not covered by this document)



Summary page of what has been selected and will be installed. Click ‘Next’ to start the setup process.

  
Starting Active Directory Installation. This process could take 10-15 minutes depending on the speed of the machine being used.

  
Once the Active Directory Installation is complete, the Wizard will start securing the communication ports and files used during domain communication.

You may be asked for the Windows 2003 Install disk at this point. If it requests it make sure it is available or you will need to install and configure DNS manually later on.



You must Setup the server with a Static IP Address in order for the server to become a domain controller. This can be any private range IP address but it is recommended that the address is easy to remember due to the frequency of use during testing. (e.g. 10.10.0.1)



Assign a Static IP address and DNS server address to Private Network Interface



This step configures DNS on the server and should take only a couple of minutes.



Click ‘Finish’ to complete the Active Directory Installation Process.



Once complete you will need to restart the computer for the changes to take effect. Click ‘Restart Now’

  
Confirmation Screen- Click ‘Finish’ to move onto the next step.

  
Click ‘Add or remove a role’ to proceed to the next step.



  
The 2003 server is now a Domain controller and a DNS server. The Next step is to setup DHCP server. If you private network already has a DHCP server on it, you may skip this next section.

## Install and Configure DHCP

  
Select ‘DHCP Server’ to add the new server role to the machine and click ‘Next’

  
Click ‘Next’ to start the New Scope Wizard.

  
Click ‘Next’ to continue

  
Give the scope a name. The name given here does not affect the configuration or how address requests are serviced.

  
Assign a range of IP address to be given out. A minimum of 10 address and a maximum of 245 should be used for the scope.



The Exclusion range is a subset of the scope that was just created. If your scope does not contain all addresses of the subnet, you do not need to set aside exclusions for Static IP assignments. Additionally, this problem can be circumvented later by making reservations for devices that require static IP addresses.



For Test Purposes, the default setting of 8 Days is fine.

  
Click ‘Next’ to continue.



Setting this option will give clients who obtain an IP address from this server a gateway other than the IP address of the domain controller. For our test environment this is not used.   
Click ‘Next’

  
The parent domain name was specified during Active Directory-DNS setup. Server name and IP address were specified during operating system installation. Use the respective names that you previously assigned. This information can be found on the Computer Name tab under System Properties if you are unsure.

  
If you have installed or plan on using WINS you may specify the server name and IP here. The Windows Mobile Test Environment does not utilize WINS.

  
Click ‘Next’ to activate the scope.

  
Click ‘Finish’ to complete the Scope Wizard.

  
Confirmation Screen shows what was just done, Click ‘Finish’ to move on to the next step.

  
 You must Authorize the Scope in Active Directory before DHCP requests will be serviced. The DHCP Management console can be accessed from the start Menu under Administrative Tools.



DHCP scope Authorized and Active. (If the scope does not show Active after a few seconds, click the ‘Refresh’ Icon on the toolbar or hit F5 on the keyboard)

## Installing Internet Information Services (IIS)

The Next Server Role to add is the Application Server Role which contains Internet Information Services (IIS) which is required for certificate services to hand out certificates.  
  
Click ‘Next’ to continue with the Wizard.  
  
Click ‘Next’ to start the install.  
  
Configuring Internet Information Services (IIS) and related components. This should take just a couple of minutes.  
  
Installing and Configuring the Server.  
  
IIS Server role has now completed installation, Click ‘Finish’

## Setting up the Internet Authentication Service (IAS).

  
Select Networking Service and Click the ‘Details’ button for a list of available Networking Services..  


While you are adding the Internet Authentication service, also add Simple TCP/IP services. This will give you the **Echo service** that is required by several of the WiFi Metrics Tests and other CETK/ LTK tests. Click ‘Ok’ to start the Installation Wizard.

  
Click ‘Finish’ to close the Installation Wizard.  
  
Internet Authentication Service must be registered in Active Directory. The IAS Management console can be accessed from the Start Menu under ‘Administrative Tools’ at any time.  
  
Click ‘Ok’ to proceed with Authorization.

## Setting up RADIUS clients

  
Once the IAS Service has completed installation a link to the IAS console will be placed under administrative tools. Clients may be added to the server by right clicking RADIUS clients in the Internet Authentication Service console.   
  
  
Create Radius Client for Access points and other RAS Clients.



Enter a shared key (The example is using 0123456789) The same key must be configured on the Access point to allow the server to communicate the certificate information to and from the client. (Note: this is assuming the default RADIUS ports of 1812 and 1813 on each side)

## Installing and Configuring Certificate Services



Select Certificate Service and Click the ‘Details’ button.  
  
  
  
Select both Certificate Services CA and Certificate Services Web Enrollment Support. Click ‘OK’  
  
No Machine or Domain name changes allowed after this point. Any necessary changes will require you to start the machine installation over again after selecting ‘Yes’.  
  
  
  
To install the Web enrollment portion of Certificate Services, IIS must be stopped and restarted.  
  
  
  
Choose Enterprise root CA and Click ‘Next’



Choose the root name of the domain upon which this server is installed.



Accept the default Certificate log locations, Click ‘Next’

  
Click ‘Finish’ to complete the Wizard.  
  


## Enabling Certificate Templates

  
From the Run line enter ‘certtmpl.msc’ and click ‘OK’

  
Enroll Permissions must be allowed prior to Installing. Click ‘Ok’

Once the Certificate Template has been installed, Open the Certification Authority Console from the Administrative tools menu.  


Right click ‘Certificate Template’, Select New Certificate Template to Issue.



Select All Certificate Templates (To Select All Shift-Click first entry and then last entry) click ‘OK’

## Setting up user groups and accounts.

  
Create the following user groups using the template above:  
- EAP-TLS Users  
- EAP-PEAP Users  
- EAP-MD5 Users

  
Create the following user accounts and passwords:

Username: eaptls Password: eaptls  
Username: eappeap Password: eappeap   
Username: eapmd5 Password: eapmd5  
  
Configure each user account with the settings below.  


After creating users, set user account properties (Right click “Properties”)   
  
Check the ‘Store Password using reversible encryption’ in account options.



Click the ‘Dial in’ tab and select allow access.  
  
Once all the user groups and accounts have been created, add the users to their respective groups by right clicking the group and typing the username in the object name box. Click ‘OK’

## Setting Remote Access Policies in IAS

  
Open IAS Management Console, Click Action Menu and ‘New Remote Access Policy’ Select ‘Wireless’ and click the ‘Next’  
  
Choose Group and Click ‘Add’  
  
Select the appropriate User Group for this policy.   
  
  
Select either Smart card or Protected EAP for the policy type. Click ‘Next’  
  
  
Click ‘Finish’ to complete the RAS Policy Wizard. Repeat the Policy Procedure for each User group previously created. (Eaptls Users, Eappeap Users, EapMd5 Users) Note: the Policy is assigned to the Group and not the individual user.

Once the Server has rebooted your Active Directory Users and Computers management console should look something like this. You may want to do a couple of manual tests on the server to make sure that it is handing out IP addresses and Certificates before attempting any Automated tests. If Everything works as expected, your RADIUS server is ready for use.

Note: a User Certificate can be manually requested through a web browser by typing “http://<Server IP Address>/certsrv” and entering the credentials for the certificate type being requested. This will only work when the RADIUS server Setup has concluded.

1. (A)utomated, (M)annual, (T)actics, (C)ETK, (L)TK, (P)roposed [↑](#footnote-ref-1)