TEST PLAN

# TP-181
# CWMP Interoperability and Functionality Test Plan

**Issue: 1 Corrigendum 1**
**Issue Date: September 2020**

**Issue History**

| Issue Number | Approval Date | Publication Date | Issue Editor | Changes |
|---|---|---|---|---|
| 1 | 25 January 2019 | 25 January 2019 | Tim Winters, UNH-IOL Marion Dillon, UNH-IOL | Original |
| Corrigendum 1 | 22 September 2020 | 22 September 2020 | Jason Walls, QA Cafe | Clarifications to tests 5.1.8, 5.5.1, 5.5.2, and incorrect parameter name (DestinationInterface→DestInterface) |

Comments or questions about this Broadband Forum Test Plan should be directed to info@broadband-forum.org.

| | | | |
|---|---|---|---|
| **Editors** | Marion Dillon | UNH-IOL | mdillon@iol.unh.edu |
| | Timothy Winters | UNH-IOL | twinters@iol.unh.edu |
| **Broadband User Services Work Area Directors** | John Blackford<br>Jason Walls | Arris<br>Qacafe | John.blackford@pace.com<br>jason@qacafe.com |

## Table of Contents

## List of Figures

## List of Tables

## Executive Summary

In order to ensure the continued growth of the TR-069 market and further the interoperability of the protocol, the Broadband Forum is creating a TR-069 Certification Program. Within this program, devices implementing a TR-069 management interface may be tested for their conformance to the TR-069 specification and various use cases. The TR-069 Certification Program started with the TR-069 Conformance Test Plan and now the CWMP Data Model Implementation Test Plan. To provide a consistent scope for this verification, BBF developed these test plans that are to be used by the testing agencies in the verification process.

This Internal Document provides a test plan that may be used to verify Interoperability of a CPE Device with an ACS Server through use cases.

# 1 Purpose and Scope

## 1.1 Purpose

The purpose of this document is to provide a set of use test cases, which will be used as a common testing language during Interoperability testing of a CPE Device with an ACS Server.

## 1.2 Scope

The tests detailed in this document are only intended to facilitate TR-069 interoperability use case testing. The tests in this document are limited to ACSs and CWMP enabled CPE devices.

## 1.3 Test Execution

The tests detailed in this document are to be run in a controlled environment. There is no specified order to the tests in this document.

# 2 References and Terminology

## 2.1 Conventions

In this Test Plan, several words are used to signify the requirements of the specification. These words are always capitalized. More information can be found be in RFC 2119.

| | |
|---|---|
| **MUST** | This word, or the terms "REQUIRED", means that the definition is an absolute requirement of the specification. |
| **MUST NOT** | This phrase means that the definition is an absolute prohibition of the specification. |
| **SHOULD** | This word, or the adjective "RECOMMENDED", means that there could exist valid reasons in particular circumstances to ignore this item, but the full implications need to be understood and carefully weighed before choosing a different course. |
| **SHOULD NOT** | This phrase, or the phrase "NOT RECOMMENDED" means that there could exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications need to be understood and the case carefully weighed before implementing any behavior described with this label. |
| **MAY** | This word, or the adjective "OPTIONAL", means that this item is one of an allowed set of alternatives. An implementation that does not include this option MUST be prepared to inter-operate with another implementation that does include the option. |

## 2.2 References

The following references are of relevance to this Test Plan. At the time of publication, the editions indicated were valid. All references are subject to revision; users of this Test Plan are therefore encouraged to investigate the possibility of applying the most recent edition of the references listed below, including any released amendments or corrigendum materials.
A list of currently valid Broadband Forum Technical Reports is published at www.broadband-forum.org.

| Document | | Title | Source | Year |
|---|---|---|---|---|
| [1] | IR-069i2 | *TR-069 Conformance Test Plan* | BBF | 2016 |
| [2] | OD-361 | *CWMP Certification Program Guidelines* | BBF | 2016 |
| [3] | TR-069 | *CPE WAN Management Protocol* | BBF | 2013 |
| [4] | TR-098 | *Internet Gateway Device Data Model for TR-069* | BBF | 2008 |
| [5] | TR-104 | *Provisioning Parameters for VoIP CPE* | BBF | 2014 |

|  |  |  |  |  |
|---|---|---|---|---|
|  | Issue 2 |  |  |  |
| [6] | TR-135 | *Data Model for a TR-069 Enabled STB* | BBF | 2012 |
| [7] | TR-140 | *TR-069 Data Model for Storage Service Enabled Devices* | BBF | 2010 |
| [8] | TR-143 | *Enabling Network Throughput Performance Test and Statistical Monitoring* | BBF | 2008 |
| [9] | TR-157 | *Component Objects for CWMP* | BBF | 2011 |
| [10] | TR-181 | *Device Data Model for TR-069* | BBF |  |
| [11] | TR-196 Issue 2 | *Femto Access Point Service Data Model* | BBF | 2011 |
| [12] | RFC 2119 | *Key words for use in RFCs to Indicate Requirement Levels* | IETF | 1997 |
| [13] | RFC 2131 | *Dynamic Host Configuration Protocol* | IETF | 1997 |
| [14] | RFC 2132 | *DHCP Options and BOOTP Vendor Extensions* | IETF | 1997 |
| [15] | RFC 3315 | *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)* | IETF | 2003 |
| [16] | RFC 5246 | *The Transport Layer Security (TLS) Protocol* | IETF | 2008 |
| [17] | RFC 5389 | *Session Traversal Utilities for NAT (STUN)* | IETF | 2008 |
| [18] | NOTE-SOAP-20000508 | *Simple Object Access Protocol (SOAP) 1.* | W3C | 2000 |
| [19] | REC-xml | *Extensible Markup Language (XML) 1.0 (Fifth Edition)* | W3C | 2008 |

## 2.3 Abbreviations

This Test Plan uses the following abbreviations:

|  |  |
|---|---|
| ACS | Auto-Configuration Server |
| CN | Common Name |
| CPE | Customer Premise Equipment |
| CWMP | CPE WAN Management Protocol |
| DHCP | Dynamic Host Configuration Protocol |
| DNS | Domain Name System |
| DSLAM | DSL Access Multiplexer |

| | |
|---|---|
| DUT | Device Under Test |
| FTP | File transfer Protocol |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol over Secure Socket Layer |
| ID | Identifier |
| IP | Internet Protocol |
| IPv6 | Internet Protocol version 6 |
| LAN | Local Area Network |
| NAT | Network Address Translation |
| NTP | Network Time Protocol |
| RFC | Request for Proposal |
| RPC | Remote Procedure Call |
| SOAP | Simple Object Access Protocol |
| SSL | Secure Socket Layer |
| STB | Set Top Box |
| STUN | Session Traversal Utilities for NAT |
| TCP | Transmission Control Protocol |
| TFTP | Tiny File transfer Protocol |
| TLS | Transport Layer Security |
| TR | Technical Report |
| TTL | Time to Live |
| UDP | User Datagram Protocol, |
| URL | Universal Resource Locator |
| URN | Uniform Resource Name |
| UTC | Coordinated Universal Time |
| UTF | Universal Multiple-Octet Coded Character Set Transformation Format |
| UUID | Universally Unique Identifier |
| VoIP | Voice over Internet Protocol |
| WAN | Wide Area Network |
| XML | Extensible Markup Language |
| TR | Technical Report |
| WA | Work Area |

# 3 Test Plan Impact

## 3.1 Energy Efficiency

TP-181 has no impact on Energy Efficiency.

## 3.2 IPv6

TP-181 intends to support the IPv6 protocol. However, in this issue of the Working Text, IPv6 specific test cases have not been identified.

## 3.3 Security

TP-181 requires the use of authentication when connecting CPE to an ACS. Use of an HTTPS URL indicates that the CPE can establish an SSL or TLS connection to the ACS.

# 4 Test Setup

## 4.1 Test Equipment

The necessary set of test equipment to deliver reliable and repeatable test results is specified in Table 4-1.

**Table 4-1: Required Test Equipment**

| **Test Equipment** | **Description and Functional Capabilities** |
|---|---|
| ACS | The ACS is an interoperability partner in this test plan. |
| Traffic Analyzer | To verify certain test metrics, Traffic Analyzers must be present between the CPE and ACS and between the CPE and any LAN Device. |
| Access Node | For infrastructure needs, an Access Node will be provided to bridge connections between the CPE and ACS. |
| File Server | The File Server must support HTTP PUT & GET, as well as FTP PUT & GET. |
| CPE | The CPE is an interoperability partner in this test plan. |
| Network Router | There may be one or more Network Routers between the CPE & ACS. If the CPE is a LAN CPE, one Network Router may be a natted device. |
| Firewall | The Firewall must support the ability to selectively block traffic based on IP address. |
| NTP Server | Two NTP Servers. Public NTP Servers such as NIST Internet Time Servers may be used. |
| RADIUS Server | To authenticate end devices in WPA Enterprise test cases |
| DHCP Server | To assign addresses and other provisioning information to the CPE on the WAN |
| DNS Server | To resolve addresses on the WAN |
| LAN Device | To connect to the LAN side of a Gateway CPE. |
| Wireless LAN Device | To connect to the LAN side of a Gateway CPE over wireless. |

## 4.2 Test Setup and Execution

The Interoperability test setup is shown below in Figure 1. The following pieces of test equipment must be connected to the Internet and reachable over IP, but there is no other topology requirement: ACS, DHCP Server, DNS Server, File Server, NTP Server, RADIUS Server. The Access Node may not be required if the primary CPE connection is Ethernet.

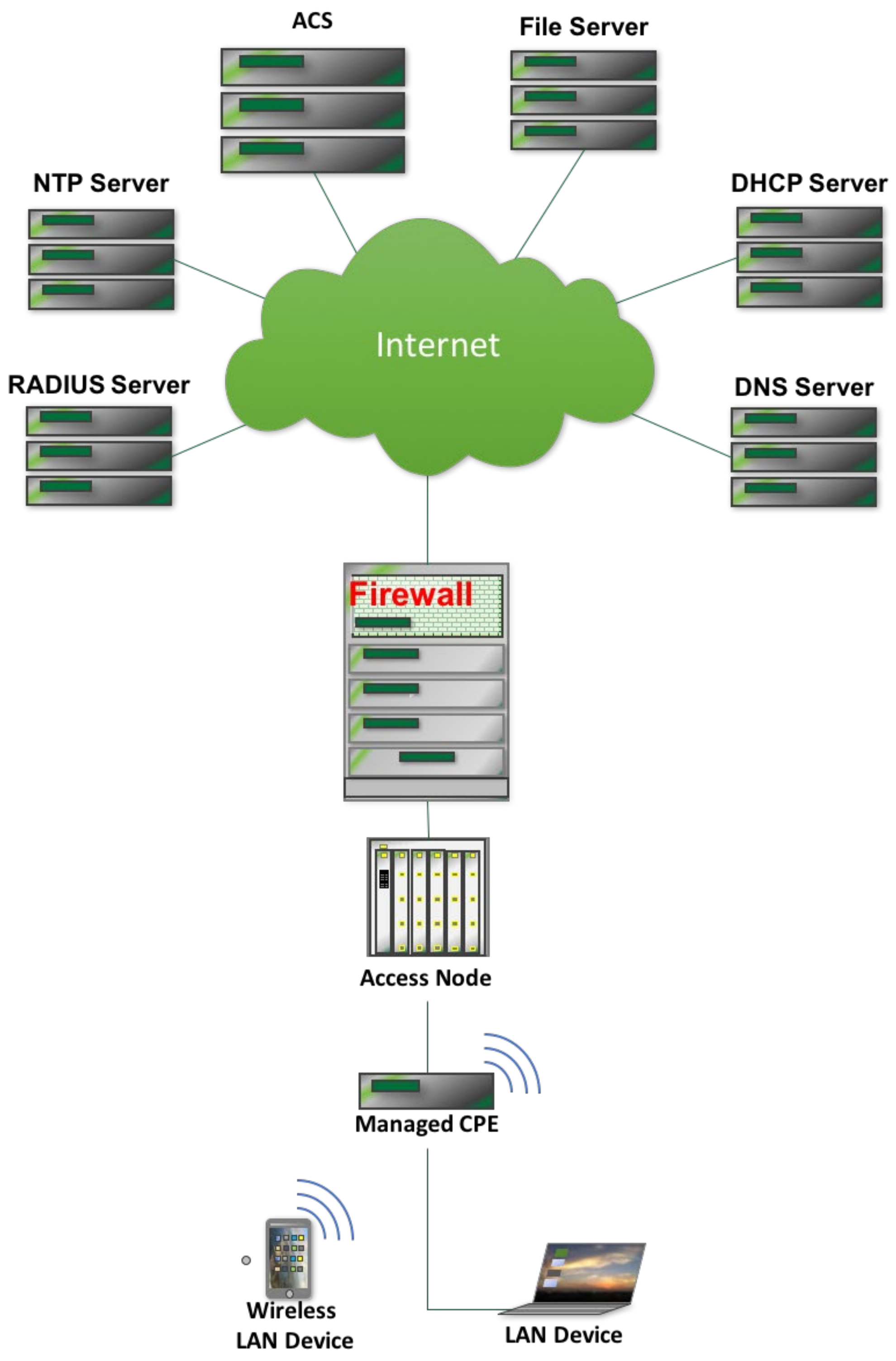

**Figure 1 - Common Topology for Interoperability Testing**

### 4.2.1 Common Test Setup

This section describes the test setup shared between all test cases. Any additional setup steps will be described in the "Test Setup" section of the test case.

1. ACS is connected to the network
2. CPE is connected to the network and configured with an ACS URL that corresponds to the ACS in step 1.
3. Have a Network Analyzer to capture traffic between ACS and CPE.

### 4.2.2 Determine WAN Interface

This section describes steps to determine the WAN interface of a device. These steps are referenced in the Test Procedures.

For CPE that support the Device:2 data model:

a. The ACS performs a GetParameterNames RPC on the Interface table. The interface with the WAN IP address is the WAN interface.

For CPE that support the InternetGatewayDevice:1 root data model:

b. The ACS performs a GetParameterValues RPC on InternetGatewayDevice.Layer3Forwarding.DefaultConnectionService. This will return the WAN interface.

### 4.2.3 Test Execution

Each test is defined as a separate entity that can be run independent of all other test procedures. These tests, performed sequentially, may cause changes to the ACS & CPE states during the course of testing.

**Note:** If a CPE returns a status of 1 in a SetParameterValuesResponse or an AddObjectResponse, the following steps must be followed:

1. Terminate the CWMP session.
2. Configure the ACS to issue a connection request.
3. Configure the ACS to issue a GetParameterValues RPC for the changed variable(s) and verify that they are correct.

## 4.3 CPE Test Requirements and Prerequisites

1. OD-361 [2] section 3.4.4 states that passing IR-069i2 [1] is a prerequisite for a CPE undergoing this test plan.
2. Each test case includes a References section that refers to a version of a data model definition or other standard document. For all Broadband Forum data models and Broadband Forum Technical Reports, the test case references the earliest version the CPE must support to run the test case.
3. Each test case includes a Profiles section that includes the profiles needed to run the test case. Any additional requirements are included in the Optional Features section. If the CPE supports the profile and the optional feature listed, the test case must be run.
    a. **Note:** For test cases that require Baseline:1 or Baseline:2 support, the CPE must support each parameter listed in the test case, but does not need to support each parameter in the profile.

b. A list of test cases by profile is included below. Note, there may be additional parameters required or required parameter values. Refer to the test case for complete information.

| Device:2 Profile | Applicable Tests |
|---|---|
| None | 5.1.5 GetRPCMethods<br>5.1.10 Encrypted Connection |
| Baseline:1 | 5.1.1 Factory Reset<br>5.1.4 Firmware Download<br>5.1.6 Configuration Backup and Restoration<br>5.1.9 SPV on a Boolean Parameter<br>5.7.1 Non-ASCII Characters in a SetParameterValues RPC<br>5.7.2 Multi-Byte Encoding in a SetParameterValues RPC<br>5.7.3 Non-ASCII Characters in a ParameterKey<br>5.7.4 Multi-Byte Encoding in a ParameterKey<br>5.7.5 Non-ASCII Characters in CommandKey<br>5.7.6 Multi-Byte Encoding in CommandKey |
| Time:1 | 5.1.2 Time Setting |
| PPPInterface:1 | 5.1.7 PPP Interface Change |
| DHCPv4Server:1 | 5.1.8 DHCP Provisioning |
| Vendor specific parameters | 5.1.11 GetParameterNames Vendor Specific Parameters<br>5.1.12 SetParameterValues Vendor Specific Parameters |
| IPPing:1 OR IPPingDetailed:1 | 5.2.1 Diagnostics IPPing |
| Download:1 | 5.2.2 Download Diagnostics – HTTP<br>5.2.3 Download Diagnostics - FTP |
| Upload:1 | 5.2.4 Upload Diagnostics – HTTP<br>5.2.4 Upload Diagnostics – FTP |
| TraceRoute:1 | 5.2.6 TraceRoute |
| UDPEcho:1 | 5.2.7 UDPEcho |
| Baseline:2, IPInterface:1, EtherenetInterface:1, WiFiSSID:1, WiFiRadio:1 | 5.3.1 Current Interface Configuration |
| Hosts:2 | 5.3.3 Connected LAN Devices - WiFi |
| Hosts:2, DHCPv4ServerClientInfo:1, DHCPv6ServerClientInfo | 5.3.4 Connected LAN Devices - DHCP |
| Hosts:1 | 5.3.5 Device Connect/Disconnect Notification Test |
| NAT:1 | 5.4.1 Create a Port Mapping – Single Interface<br>5.4.2 Create a Port Mapping – All Interfaces<br>5.4.3 Create a Port Mapping – External Port |

| | |
|---|---|
| | Range<br>5.4.4 Create a Port Mapping – Lease Duration > 0<br>5.4.5 Create a Port Mapping – Remote Host Restriction<br>5.4.6 Create a Port Mapping – Multiple Entries – Precedence Rules<br>5.4.7 Modify a Port Mapping<br>5.4.8 Delete a Port Mapping<br>5.4.9 Create a Port Mapping – TCP |
| AdvancedFirewall:1 | 5.5.1 Default Policy<br>5.5.2 Deny/Allow Outbound Protocols<br>5.5.3 Deny/Allow Outbound Ports<br>5.5.4 Deny/Allow Source IP Address |
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | 5.6.6 Wi-Fi Setup WEP-64<br>5.6.7 Wi-Fi Setup WEP-128<br>5.6.8 Wi-Fi Setup WPA Personal<br>5.6.9 Wi-Fi Setup WPA Enterprise<br>5.6.10 Wi-Fi Setup WPA2 Personal<br>5.6.11 Wi-Fi Setup WPA2 Enterprise<br>5.6.12 Wi-Fi Setup WPA-WPA2 Personal<br>5.6.13 Wi-Fi Setup WPA-WPA2-Enterprise<br>5.6.14 Wi-Fi Setup – Add SSID<br>5.6.15 Wi-Fi Setup – Remove SSID |

| **InternetGatewayDevice:2 Profile** | **Applicable Tests** |
|---|---|
| None | 5.1.5 GetRPCMethods<br>5.1.10 Encrypted Connection |
| Baseline:1 | 5.1.1 Factory Reset<br>5.1.4 Firmware Download<br>5.1.6 Configuration Backup and Restoration<br>5.1.9 SPV on a Boolean Parameter<br>5.3.5 Device Connect/Disconnect Notification Test<br>5.4.1 Create a Port Mapping – Single Interface<br>5.4.3 Create a Port Mapping – External Port Range<br>5.4.4 Create a Port Mapping – Lease Duration > 0<br>5.4.5 Create a Port Mapping – Remote Host Restriction<br>5.4.6 Create a Port Mapping – Multiple Entries – Precedence Rules<br>5.4.7 Modify a Port Mapping |

| | |
|---|---|
| | 5.4.8 Delete a Port Mapping<br>5.4.9 Create a Port Mapping – TCP |
| Time:2 | 5.1.3 Time Setting |
| Baseline:2 | 5.1.7 PPP Interface Change<br>5.3.2 Connected LAN Devices<br>5.1.8 DHCP Provisioning |
| Vendor specific parameters | 5.1.11 GetParameterNames Vendor Specific Parameters<br>5.1.12 SetParameterValues Vendor Specific Parameters |
| Baseline:1, IPPing:1 | 5.2.1 Diagnostics IPPing |
| Download:1 | 5.2.2 Download Diagnostics – HTTP<br>5.2.3 Download Diagnostics - FTP |
| Upload:1 | 5.2.4 Upload Diagnostics – HTTP<br>5.2.5 Upload Diagnostics - FTTP |
| TraceRoute:1 | 5.2.6 TraceRoute |
| UDPEcho:1 | 5.2.7 UDPEcho |
| WiFiLan:1 | 5.6.1 Wi-Fi Setup WEP 64<br>5.6.2 Wi-Fi Setup WEP 128<br>5.6.3 Wi-Fi Setup WPA Personal<br>5.6.4 Wi-Fi Setup WPA2 Personal<br>5.6.5 Wi-Fi Setup WPA-WPA2 Personal |

## 4.4 ACS Test Requirements

1. The ACS MUST be configurable to include an interface that allows control of the ACS to execute the test procedures.
2. An API SHOULD be provided to the test lab to support automation of this test plan.
3. The ACS MUST allow its certificates to be configured.

## 4.5 Interoperability Testing

This test plan tests the ACS/CPE system, therefore a failure may indicate a deficiency from either the ACS or CPE.

## 4.6 Test Validation

A test is considered successful (or passed) when the corresponding test procedure has been completed and the specified success metrics are attained. Tests can be validated by observing functional changes in the DUTs, through feedback interfaces on the devices under test, results attained from the ACS, and via a Traffic Analyzer connected to the relevant links.

# 5 Test Procedures

## 5.1 Basic Setup

### 5.1.1 Factory Reset

***Purpose:***

To verify that an ACS can perform a Factory Reset on the CPE.

***References:***

InternetGatewayDevice:1.0
Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| Baseline:1 | Baseline:1 |

***Optional Features***

Factory Reset RPC

***Parameters:***

The following parameter is required to be implemented for this test. CPE supports FactoryReset RPC.

For CPE that support the Device:2 data model:

| **Device.DeviceInfo.** | |
|---|---|
| **ProvisioningCode** | <string> |

For CPE that support the InternetGatewayDevice:1 data model:

| **InternetGatewayDevice.DeviceInfo.** | |
|---|---|
| **ProvisioningCode** | <string> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

***Procedure:***

1. ACS performs a GetParameterValues RPC on the above parameter to determine its value.
2. ACS performs a SetParameterValues RPC on the above parameter for a value that is different than the value returned in step 1.
3. ACS performs a FactoryReset RPC to the CPE.
4. Allow the ACS to perform bootstrap procedures on the CPE, if it is configured to do so.
5. ACS performs a GetParameterValues RPC on the above parameter.

***Test Metrics:***

1. Validate that the FactoryReset successfully occurs via FactoryResetResponse and a "0 BOOTSTRAP" Inform RPC is sent by CPE.
2. Validate that the value of the above parameter returned in step 5 is different than it was set to in Procedure step 2.

### 5.1.2 Time Setting – Device:2 Only

***Purpose:***

To verify that an ACS can set time configuration on the CPE.

***References:***

Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Time:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

| Device.Time. | |
|---|---|
| **Enable** | True |
| **Status** | Returned from device |
| **NTPServer1** | <IP Address of NTP Server1> |
| **NTPServer2** | <IP Address of NTP Server2> |
| **CurrentLocalTime** | Returned from device |
| **LocalTimeZone** | <Local Time Zone definition in IEEE 1003.1 (POSIX) format> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have two NTP servers accessible on the WAN.
3. Have the ability to block data communications to the NTP Servers.

***Procedure:***

1. ACS performs a SetParameterValues RPC on the NTPServer1, NTPServer2, TimeZone, and Enable parameters.
2. Test setup blocks NTPServer2 address.
3. CPE is rebooted.
4. ACS performs a GetParameterValues RPC on the Status and CurrentLocalTime parameters.
5. Test setup blocks NTPServer1 address and unblocks NTPServer2 address
6. CPE is rebooted
7. ACS performs a GetParameterValues RPC on the Status and LocalTime parameters.

***Test Metrics:***

1. Validate that the SetParameterValuesResponse is received successfully for setting both NTPServer1 and NTPServer2.
2. After the reboot, validate that the Traffic Analyzer shows CPE communication with NTPServer1.
3. Validate that the time is set on the CPE via the GetParameterValuesResponse.
4. After the reboot, validate that the Traffic Analyzer shows CPE communication with NTPServer2.
5. Validate that the time is set on the CPE via the GetParameterValuesResponse.

### 5.1.3 Time Setting – InternetGatewayDevice:1 Only

***Purpose:***

To verify that an ACS can set time configuration on the CPE.

***References:***

InternetGatewayDevice:1.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| N/A | Time:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

| **InternetGatewayDevice.Time.** | |
|---|---|
| **NTPServer1** | <IP Address of NTP Server1> |
| **NTPServer2** | <IP Address of NTP Server2> |
| **CurrentLocalTime** | Returned from device |
| **LocalTimeZone** | <Time Zone offset> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have two NTP servers accessible on the WAN.
3. Have the ability to block data communications to the NTP Servers.

***Procedure:***

1. ACS performs a SetParameterValues RPC on the NTPServer1, NTPServer2, and TimeZone parameters.
2. Test setup blocks NTPServer2 address.
3. CPE is rebooted.
4. ACS performs a GetParameterValues RPC on the LocalTime parameters.
5. Test setup blocks NTPServer1 address and unblocks NTPServer2 address
6. CPE is rebooted
7. ACS performs a GetParameterValues RPC on the CurrentLocalTime parameter.

***Test Metrics:***

1. Validate that the SetParameterValuesResponse is received successfully for setting both NTPServer1 and NTPServer2.

2. After the reboot, validate that the Traffic Analyzer shows CPE communication with NTPServer1.
3. Validate that the time is set on the CPE via the GetParameterValuesResponse.
4. After the reboot, validate that the Traffic Analyzer shows CPE communication with NTPServer2.
5. Validate that the time is set on the CPE via the GetParameterValuesResponse.

### 5.1.4 Firmware Download

***Purpose:***

To verify that an ACS can perform a firmware download on the CPE.

***References:***

InternetGatewayDevice:1.0
Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Baseline:1 | Baseline:1 |

***Optional Features***

None

***Parameters:***

The following parameter is required to be implemented by the CPE for this test.

For CPE that support the Device:2 data model:

| Device.DeviceInfo. | |
|---|---|
| **SoftwareVersion** | Returned from device |

For CPE that support the InternetGatewayDevice:1 data model:

| InternetGatewayDevice.DeviceInfo. | |
|---|---|
| **SoftwareVersion** | Returned from device |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a file server accessible on the WAN with known credentials.
3. Place firmware to download on the file server.

***Procedure:***

1. ACS issues a GetParameterValues RPC on the CPE SoftwareVersion parameter listed in the parameters section to learn the CPEs current software version
2. ACS issues a Download RPC to the CPE with the following arguments set

| Download RPC Argument | Value |
|---|---|
| CommandKey | <string> |
| FileType | "1 Firmware Upgrade Image" |
| URL | URL of firmware image on the file server |
| Username | File server username |
| Password | File server password |

| | |
|---|---|
| FileSize | Size of file to download in bytes |
| TargetFileName | Name of the firmware to download |
| DelaySeconds | 1 |
| SuccessURL | <empty> |
| FailureURL | <empty> |

3. Allow the CPE to complete firmware download from the file server, apply the firmware, and reboot if necessary
4. After CPE issues a “7 TRANSER COMPLETE” event code, the ACS issues a GetParameterValues RPC on the on the CPE SoftwareVersion parameter listed in the parameters section to learn the CPEs new software version

***Test Metrics:***

1. Ensure the software version reported by the CPE before the firmware download is different than the software version reported by the CPE after the firmware download.

### 5.1.5 GetRPCMethods

***Purpose:***

This test is designed to verify that the ACS can schedule a GetRPCMethods RPC to a CPE and get back the correct response

***References:***

Section A.3.1.1/TR-069

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| N/A | N/A |

***Optional Features***

None

***Parameters:***

None.

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

***Procedure:***

1. ACS schedules a GetRPCMethods RPC on the CPE.
2. Allow the CPE to respond with the GetRPCMethodsResponse

***Test Metrics:***

1. Verify that the CPE responds to the GetRPCMethods RPC
2. Verify that the GetRPCMethodsResponse contains all mandatory RPCs

### 5.1.6 Configuration Backup and Restoration

***Purpose:***

This test is designed to verify that the ACS can schedule an upload of a CPE's configuration and then perform a restoration of that same configuration.

***References:***

Section A.4.1.5/TR-069a1 or later

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Baseline:1 | Baseline:1 |

***Optional Features***

Upload RPC

***Parameters:***

These arguments are for the Upload and Download RPCs. Note, if the CPE supports Upload FileType "3 Vendor Configuration File <i>", the test case MUST be run using that FileType.

| Upload RPC Argument | Value |
|---|---|
| CommandKey | <string> |
| FileType | "3 Vendor Configuration File <i>"<br>OR<br>"1 Vendor Configuration File" |
| URL | URL of firmware image on the file server |
| Username | File server username |
| Password | File server password |
| DelaySeconds | 0 |

| Download RPC Argument | Value |
|---|---|
| CommandKey | <string> |
| FileType | "3 Vendor Configuration File" |
| URL | URL of configuration file on the file server |
| Username | File server username |
| Password | File server password |
| FileSize | Size of file to download in bytes |
| TargetFileName | Name of the configuration file to download |
| DelaySeconds | 0 |

| | |
|---|---|
| SuccessURL | <empty> |
| FailureURL | <empty> |

The following parameter path is required to be implemented for this test.

For CPE that support the Device:2 data model:

**Device.DeviceInfo.VendorConfigFile.**

For CPE that support the InternetGatewayDevice:1 data model:

**InternetGatewayDevice. DeviceInfo.VendorConfigFile.**

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a file server accessible on the WAN with known credentials that can accept uploads and downloads.
3. If CPE supports "3 Vendor Configuration File <i>" Upload FileType, perform a GetParameterValues RPC on the partial path above to determine the correct instance number to upload.

***Procedure:***

1. ACS performs a GetParameterValues RPC on PeriodicInformInterval to determine its value.
2. ACS schedules an Upload RPC on the CPE with the parameters above.
3. Allow the upload to complete.
4. ACS performs a SetParameterValues RPC on PeriodicInformInterval for a value that is different than the value returned in step 1.
5. Wait for the CPE to send 2 Periodic Informs and check the interval between them.
6. ACS schedules a download RPC on the CPE with the parameters above.
7. Allow the download to complete.
8. Wait for the CPE to send 2 Periodic Informs and check the interval between them.

***Test Metrics:***

1. Verify that the CPE responds to the Upload RPC with an UploadResponse
2. Verify that the DUT uploaded the configuration to the server
3. Verify that the CPE sends Periodic Informs at the interval set in Step 4.
4. Verify that the CPE responds to the Download RPC with a DownloadResponse
5. Verify that the CPE sends Periodic Informs at the interval returned from the device in Step 1.

### 5.1.7 PPP Interface Change

***Purpose:***

This test is designed to verify that the ACS can configure the PPP interface of the CPE.

***References:***

InternetGatewayDevice:1.4
Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| PPPInterface:1 | Baseline:2 |

***Optional Features***

None

***Parameters:***

This test is only applicable to CPE that use PPP to obtain an IP address. The following parameters are required to be implemented for this test.

For CPE that support the Device:2 data model:

| Device.PPP.Interface.{i}. | |
|---|---|
| **Username** | username1, username2 |
| **Password** | password1, password2 |
| **Reset** | true |

For CPE that support the InternetGatewayDevice:1 data model:

| InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.{i}. | |
|---|---|
| **Username** | username1, username2 |
| **Password** | password1, password2 |
| **Reset** | true |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Setup two valid sets of credentials on the PPP server, username1/password1 and username2/password2

***Procedure:***

1. ACS determines the instance number of the PPP interface.
2. ACS performs a GetParameterValues RPC on the CPE for Username and Password.
3. Allow the CPE to respond with a GetParameterValuesResponse
4. ACS performs a SetParameterValues RPC on the CPE for Username, Password, and Reset, using the second set of valid credentials, username2/password2.

5. Allow the CPE to respond with a SetParameterValuesResponse and establish a new PPP session.
6. ACS performs a GetParameterValues RPC on the CPE for Username and Password.
7. Allow the CPE to respond with a GetParameterValuesResponse.
8. Reboot the device.
9. ACS performs a GetParameterValues RPC on the CPE for Username and Password.

***Test Metrics:***

1. Validate that the CPE sends a GetParameterValuesResponse in step 3 containing Username=username1 and Password=<empty string>.
2. Validate that the CPE sends a SetParameterValuesResponse in step 5 with Status=1 and that the CWMP session is successfully ended.
3. Validate that the CPE successfully establishes a new PPP session using the new credentials.
4. Validate that the CPE sends a GetParameterValuesResponse in step 7 containing Username=username2 and Password=<empty string>.
5. Validate that the CPE sends a GetParameterValuesResponse in step 9 containing Username=username2 and Password=<empty string>.

### 5.1.8 DHCP Provisioning

***Purpose:***

To verify that an ACS can configure the basic DHCPv4 configuration required on a CPE device.

***References:***

InternetGatewayDevice:1.0
Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| DHCPv4Server:1 | Baseline:2 |

***Optional Features***

DHCPLeaseTime is included in data model for CPE that support the InternetGatewayDevice:1 data model.

***Parameters:***

The following parameters are required to be implemented for this test.

For CPE that support the Device:2 root data model:

| **Device.DHCPv4.Server.Pool.{i}.** | |
|---|---|
| **Enable** | TRUE |
| **Interface** | Returned from device |
| **MinAddress** | <minimum address> |
| **MaxAddress** | <maximum address> |
| **LeaseTime** | 60 |

For CPE that support the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.LANDevice.{i}.LANHostConfigManagement.** | |
|---|---|
| **DHCPServerConfigurable** | TRUE |
| **DHCPServerEnable** | TRUE |
| **MinAddress** | <minimum address> |
| **MaxAddress** | <maximum address> |
| **DHCPLeaseTime** | 60 |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. CPE supports the appropriate DHCP parameters in the parameters section
3. Have a LAN device that can connect to the CPE via DHCP.
4. Have a Network Analyzer to capture traffic between the CPE and LAN device.
5. If the DUT implements the Device:2 root data model, determine whether or not an existing entry exists for the object specified above.

***Procedure:***

1. If the DUT implements the Device:2 root data model, and no entry exists for Device.DHCPv4.Server.Pool.{i}., perform an AddObject RPC on Device.Server.Pool., recording the InstanceNumber value returned in the AddObjectResponse.
2. On the ACS schedule a SetParameterValues RPC on the appropriate Configurable parameters in the Parameters Section, setting the DHCP pool's minimum and maximum to be completely outside the existing range, if any (for example, by setting the new minimum above the current maximum).
3. Allow the end system to connect to the CPE via DHCP.
4. Perform a reboot of the CPE
5. Allow the end system to connect to the CPE via DHCP

***Test Metrics:***

1. Verify that the SetParameterValuesResponse RPC is correct for all parameters
2. Verify that the end system connects to the CPE within the designated range.
3. Verify that that end system connects to the CPE within the designated range after the reboot

### 5.1.9 SPV on a Boolean Parameter

***Purpose:***

This is a test case to verify that the ACS can change the Boolean on a CPE and receive the correct response.

***References:***

Section A.3.2.1/TR-069a1 or later

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Baseline:1 | Baseline:1 |

***Optional Features***

None

***Parameters:***

Writable boolean parameter from the device's data model.

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

***Procedure:***

1. ACS establishes a session
2. ACS performs a GetParameterValues RPC on the CPE on a Boolean parameter.
3. ACS schedules a SetParameterValues RPC on the CPE, to change the Boolean parameter to the opposite of the value returned in step #2.
4. Allow the CPE to respond with the SetParameterValuesResponse.
5. ACS schedules a GetParameterValues RPC on the CPE, to request the Boolean parameter.
6. Allow the CPE to respond with the GetParameterValuesResponse

***Test Metrics:***

1. Verify that the CPE responds to the SetParameterValues RPC
2. Verify that the SetParameterValuesResponse contains status 0.
3. Verify that the GetParameterValuesResponse contains the changed value.

### 5.1.10 Encrypted Connection

***Purpose:***

To verify that an ACS and CPE can interoperate using Encryption protocols.

***References:***

Section 3.3/TR-069a1

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| N/A | N/A |

***Optional Features***

Secure CWMP

***Parameters:***

Readable Parameter from the device's data model.

***Test Setup:***

1. Load the appropriate Certificate of Authentication onto the CPE.
2. Have the private portion of the certificate on the ACS.
3. Refer to Common Test Setup for setup steps.

***Procedure:***

1. Allow the CPE to establish an encrypted CWMP session with the ACS.
2. On the ACS schedule a GetParametersValues RPC on the CPE.
3. Allow the CPE to respond with the GetParametersValuesResponse containing the requested parameters.

***Test Metrics:***

1. Verify that the CWMP session is encrypted.
2. Verify that the GetParameterValuesResponse is valid.

### 5.1.11 GetParameterNames Vendor Specific Parameters

***Purpose:***

To verify that vendor specific CPE objects and parameters can be read and processed in an ACS.

***References:***

Vendor supplied vendor-specific parameters

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Vendor specific parameters | Vendor specific parameters |

***Optional Features***

None

***Parameters:***

Vendor supplied vendor specific parameter, type, and purpose. Vendor supplied vendor specific object and purpose.

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Obtain a list of vendor specific parameters from the vendor

***Procedure:***

1. Allow the CPE to establish an CWMP session with the ACS
2. Schedule a GetParameterNames RPC on the CPE entire data model.

***Test Metrics:***

1. Verify that the ACS can process all vendor specific parameters and objects.

### 5.1.12 SetParameterValues Vendor Specific Parameters

***Purpose:***

To verify that vendor specific CPE parameter can be written by an ACS.

***References:***

Vendor supplied technical survey containing vendor specific information.

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Vendor specific parameters | Vendor specific parameters |

***Optional Features***

None

***Parameters:***

Vendor supplied vendor-specific parameter, type, and purpose.

***Test Setup:***

2. Refer to Common Test Setup for setup steps.
3. Obtain a writable vendor specific parameter, parameter type, and parameter purpose from the vendor.

***Procedure:***

1. ACS performs a GetParameterValues RPC on the parameter in Test Setup #3.
2. ACS performs a SetParameterValues RPC on the parameter in Test Setup #3 for a value that is different than the value returned in Step 1.
3. ACS performs a GetParameterValues RPC on the parameter in Test Setup #3.
4. Reboot the device.
5. ACS performs a GetParameterValues RPC on the parameter in Test Setup #3.

***Test Metrics:***

1. Verify that the SetParameterValues RPC in Step 2 completes and the CPE responds with a correct SetParameterValuesResponse RPC.
2. Verify that the GetParametersValues RPC in Step 3 completes and the CPE responds with a correct GetParameterValuesResponse RPC.
3. Verify that the value in the GetParameterValues in Step 3 matches the value set in Step 2.
4. Verify that the value in the GetParameterValues in Step 5 matches the value set in Step 2.

## 5.2 Diagnostics

### 5.2.1 Diagnostics IPPing

***Purpose:***

To verify that an ACS can perform an IP Ping Diagnostics test on the CPE.

***References:***

InternetGatewayDevice:1.0
Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| IPPing:1 OR IPPingDetailed:1 | IPPing:1, Baseline:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test.

For CPE that implement the Device:2 root data model:

| **Device.IP.Diagnostics.IPPing.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **Host** | <IP address> |
| **NumberOfRepetitions** | 3 |
| **Timeout** | 1000 |
| **DataBlockSize** | 128 |
| **DSCP** | 0 |
| **SuccessCount** | Returned from device |
| **FailureCount** | Returned from device |
| **Device.IP.Interface.** | |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.IPPingDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **Host** | <IP address> |
| **NumberOfRepetitions** | 3 |
| **Timeout** | 1000 |
| **DataBlockSize** | 128 |
| **DSCP** | 0 |
| **SuccessCount** | Returned from device |
| **FailureCount** | Returned from device |

| **InternetGatewayDevice.Layer3Forwarding.Default.** | |
|---|---|
| **DefaultConnectionService.** | Returned from device |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a host on the WAN network that can answer pings.
3. Refer to Determine WAN Interface to determine the WAN interface. This will be used for the Interface parameter.

***Procedure:***

1. ACS performs a SetParameterValues RPC on the ping diagnostics parameters in the parameter section above.
2. ACS performs a GetParameterValues on the InternetGatewayDevice.IPPingDiagnostics or Device.IP.Diagnostics.IPPing to determine results of ping test

***Test Metrics:***

1. Verify that the CPE transmitted 3 ICMP Echo Requests to the target Host.
2. Validate that the CPE sends an Inform RPC to the ACS with Event Code “8 DIAGNOSTICS COMPLETE”.
3. Validate that the DiagnosticsState is set to “Complete”.
4. Verify that the SuccessCount and FailureCount match what is seen on the network capture.

### 5.2.2 Download Diagnostics over HTTP

***Purpose:***

To verify that an ACS and CPE can interoperate while performing the download diagnostics function over HTTP. This test will be run if supported on the CPE.

***References:***

InternetGatewayDevice.1.3
Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| Download:1 | Download:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.IP.Diagnostics.DownloadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **DownloadURL** | <URL of file on HTTP server> |
| **TestBytesReceived** | Returned from device |
| **TotalBytesReceived** | Returned from device |
| **DownloadTransports** | Returned from device |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.DownloadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **DownloadURL** | <URL of file on HTTP server> |
| **TestBytesReceived** | Returned from device |
| **TotalBytesReceived** | Returned from device |
| **InternetGatewayDevice.Capabilities.PerformanceDiagnostics.** | |
| **DownloadTransports** | Returned from device |

The DownloadTransports parameter must include HTTP for this test to be executed.

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an HTTP server and file to perform the download. The file should be big enough for the file transfer to last at least 30 seconds.

***Procedure:***

1. On the ACS schedule a SetParameterValues RPC on the Interface, DownloadURL and DiagnosticsState parameters listed in the parameters section.
2. Allow the CPE to perform the specific diagnostic tests and send an Inform message with an event code of “8 DIAGNOSTICS COMPLETE”.
3. On the ACS schedule a GetParameterValues RPC on the DownloadDiagnostics object.
4. Perform a reboot of the CPE.
5. On the ACS schedule a GetParameterValues RPC on the DownloadDiagnostics object.

***Test Metrics:***

1. Verify that the SetParameterValuesResponse RPC is correct
2. Verify that an Inform message is received with an event code of “8 DIAGNOSTICS COMPLETE”
3. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
4. Verify that TestBytesReceived is correct.
5. Verify that EOMTime – BOMTime is within 1 second of the time that the file server reports the transfer took.
6. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
    a. If the value is “Completed” verify that the values in the DownloadDiagnostic object are the same as they were before the reboot.
7. Verify that TotalBytesReceived is within 1% of the measured traffic.

### 5.2.3 Download Diagnostics over FTP

***Purpose:***

To verify that an ACS and CPE can inter-operate while performing the download diagnostics function over FTP. This test will be run if supported on the CPE.

***References:***

InternetGatewayDevice.1.3
Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| Download:1 | Download:1 |

***Optional Features***

DownloadTransports includes FTP

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.IP.Diagnostics.DownloadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **DownloadURL** | <URL of file on FTP server> |
| **TestBytesReceived** | Returned from device |
| **DownloadTransports** | Returned from device |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.DownloadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **DownloadURL** | <URL of file on FTP server> |
| **TestBytesReceived** | Returned from device |
| **TotalBytesReceived** | Returned from device |
| **InternetGatewayDevice.Capabilities.PerformanceDiagnostics.** | |
| **DownloadTransports** | Returned from device |

The DownloadTransports parameter must include FTP for this test to be executed.

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an FTP server and file to perform the download. The file should be big enough for the file transfer to last at least 30 seconds.

***Procedure:***

1. On the ACS schedule a SetParameterValues RPC on the Interface, DownloadURL and DiagnosticsState parameters listed in the parameters section.
2. Allow the CPE to perform the specific diagnostic tests and send an Inform message with an event code of “8 DIAGNOSTICS COMPLETE”.
3. On the ACS schedule a GetParameterValues RPC on the DownloadDiagnostics object.
4. Perform a reboot of the CPE.
5. On the ACS schedule a GetParameterValues RPC on the DownloadDiagnostics object.

***Test Metrics:***

1. Verify that the SetParameterValuesResponse RPC is correct
2. Verify that an Inform message is received with an event code of “8 DIAGNOSTICS COMPLETE”
3. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
4. Verify that TestBytesReceived is correct.
5. Verify that EOMTime – BOMTime is within 1 second of the time that the file server reports the transfer took.
6. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
    a. If the value is “Completed” verify that the values in the DownloadDiagnostic object are the same as they were before the reboot.

### 5.2.4 Upload Diagnostics over HTTP

***Purpose:***

To verify that an ACS and CPE can inter-operate while performing the upload diagnostics function over HTTP. This test will be run if supported on the CPE.

***References:***

InternetGatewayDevice.1.3
Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Upload:1 | Upload:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.IP.Diagnostics.UploadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **UploadURL** | <URL of HTTP server> |
| **TestFileLength** | <value long enough for the upload to last 30 seconds> |
| **TotalBytesSent** | Returned from device |
| **UploadTransports** | Returned from device |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.UploadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **UploadURL** | <URL of HTTP server> |
| **TestFileLength** | <value long enough for the upload to last 30 seconds> |
| **TotalBytesSent** | Returned from device |
| **InternetGatewayDevice.Capabilities.PerformanceDiagnostics.** | |
| **UploadTransports** | Returned from device |

The UploadTransports parameter must include HTTP for this test to be executed.

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an HTTP server to perform the Upload.
3. Wireshark running between ACS and CPE.

***Procedure:***

1. On the ACS schedule a SetParameterValues RPC on the Interface, UploadURL and DiagnosticsState parameters listed in the parameters section.
2. Allow the CPE to perform the specific diagnostic tests and send an Inform message with an event code of "8 DIAGNOSTICS COMPLETE".
3. On the ACS schedule a GetParameterValues RPC on the UploadDiagnostics object.
4. Perform a reboot of the CPE
5. On the ACS schedule a GetParameterValues RPC on the UploadDiagnostics object.

***Test Metrics:***

1. Verify that the SetParameterValuesResponse RPC is correct
2. Verify that an Inform message is received with an event code of "8 DIAGNOSTICS COMPLETE"
3. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
4. Verify that the test server received a file of TestLengthBytes size.
5. Verify that EOMTime – BOMTime is within 1 second of the time that the file server reports the transfer took.
6. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
   a. If the value is "Completed" verify that the values in the UploadDiagnostic object are the same as they were before the reboot.
7. Verify that TotalBytesSent is within 1% of the measured traffic.

### 5.2.5 Upload Diagnostics over FTP

***Purpose:***

To verify that an ACS and CPE can inter-operate while performing the upload diagnostics function over FTP. This test will be run if supported on the CPE.

***References:***

InternetGatewayDevice.1.3
Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Upload:1 | Upload:1 |

***Optional Features***

UploadTransports includes FTP

***Parameters:***

The following parameters within the InternetGatewayDevice.UploadDiagnostics table (for devices that implement the InternetGatewayDevice:1 root data model) and Device.IP.Diagnostics.UpLoadDiagnostics table (for devices that implement the Device:2 root data model) are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.IP.Diagnostics.UploadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **UploadURL** | <URL of FTP server> |
| **TestFileLength** | <value long enough for the upload to last 30 seconds> |
| **UploadTransports** | Returned from device |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.UploadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **UploadURL** | <URL of FTP server> |
| **TestFileLength** | <value long enough for the upload to last 30 seconds> |
| **InternetGatewayDevice.Capabilities.PerformanceDiagnostics.** | |
| **UploadTransports** | Returned from device |

The UploadTransports parameter must include FTP for this test to be executed.

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an FTP server to perform the Upload.

***Procedure:***

1. On the ACS schedule a SetParameterValues RPC on the Interface, UploadURL and DiagnosticsState parameters listed in the parameters section.
2. Allow the CPE to perform the specific diagnostic tests and send an Inform message with an event code of "8 DIAGNOSTICS COMPLETE".
3. On the ACS schedule a GetParameterValues RPC on the UploadDiagnostics object.
4. Perform a reboot of the CPE
5. On the ACS schedule a GetParameterValues RPC on the UploadDiagnostics object.

***Test Metrics:***

1. Verify that the SetParameterValuesResponse RPC is correct
2. Verify that an Inform message is received with an event code of "8 DIAGNOSTICS COMPLETE"
3. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
4. Verify that the test server received a file of TestLengthBytes size.
5. Verify that EOMTime – BOMTime is within 1 second of the time that the file server reports the transfer took.
6. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
   a. If the value is "Completed" verify that the values in the UploadDiagnostic object are the same as they were before the reboot.

### 5.2.6 TraceRoute

***Purpose:***

This test is designed to validate that the CPE supports TraceRoute diagnostics test and reports results appropriately to the ACS.

***References:***

InternetGatewayDevice:1.4
Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| TraceRoute:1 | TraceRoute:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.IP.Diagnostics.TraceRoute.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **Host** | <IP address> |
| **NumberOfTries** | 3 |
| **Timeout** | 5000 |
| **DataBlockSize** | 128 |
| **DSCP** | 0 |
| **MaxHopCount** | 30 |
| **Device.IP.Interface.** | |
| **Device.IP.Diagnostics.TraceRoute.RouteHops.** | |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.TraceRouteDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **Host** | <IP address> |
| **NumberOfTries** | 3 |
| **Timeout** | 5000 |
| **DataBlockSize** | 128 |
| **DSCP** | 0 |
| **MaxHopCount** | 30 |
| **InternetGatewayDevice.Layer3Forwarding.Default.** | |
| **DefaultConnectionService.** | Returned from device |

**InternetGatewayDevice.TraceRouteDiagnostics.RouteHops.**

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a host on the WAN network that can answer the CPE's traceroute mechanism.

***Procedure:***

1. Establish a CWMP session between the CWMP Analyzer and DUT with successful Inform exchanges.
2. Schedule a GetParameterValues RPC on the current WAN interface.
3. Schedule a SetParameterValues RPC on the DUT on the trace route diagnostic parameters listed above.
4. Schedule a GetParameterValues RPC on the DUT on Diagnostic State.
5. Schedule a GetParameterValues RPC on the CPE on the RouteHops table.

***Test Metrics:***

1. The DUT can properly respond to the GetParameterValues request on WAN interface.
2. The DUT is able to properly respond to the SetParameterValues for diagnostics parameter.
3. Verify that the CPE performed the TraceRoute test on the target Host.
4. Validate that the CPE sends an Inform RPC to the ACS with Event Code "8 DIAGNOSTICS COMPLETE".
5. Validate that the DiagnosticsState is set to "Complete".
6. Validate that the Entries in the RouteHops table match the traceroute traffic on the WAN.

### 5.2.7 UDPEcho Test

***Purpose:***

This test is designed to verify that the ACS can configure the UDPEcho service on the CPE and that the CPE implements that service correctly.

***References:***

InternetGatewayDevice:1.3
Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| UDPEcho:1 | UDPEcho:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.IP.Diagnostics.UDPEchoConfig** | |
|---|---|
| **Enable** | <boolean> |
| **Interface** | <empty string> |
| **SourceIPAddress** | <IP Address of UDP Client> |
| **UDPPort** | <port> |
| **PacketsReceived** | Returned from device |
| **PacketsResponded** | Returned from device |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.UDPEchoConfig.** | |
|---|---|
| **Enable** | <boolean> |
| **Interface** | <empty string> |
| **SourceIPAddress** | <IP Address of UDP Client> |
| **UDPPort** | <port> |
| **PacketsReceived** | Returned from device |
| **PacketsResponded** | Returned from device |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. UDP client available at source to send UDPEcho packets

***Procedure:***

1. ACS schedules a SetParameterValues RPC on the CPE, setting Enable to false.
2. Allow the CPE to respond with the SetParameterValuesResponse

3. Orderly terminate the session
4. Use the UDP client at source to send n UDP Echo packets to the CPE
5. ACS schedules a SetParameterValues RPC on the CPE, setting the SourceIPAddress and UDPPort that will be used, and setting Enable to true.
6. Allow the CPE to respond with the SetParameterValuesResponse
7. ACS schedules a GetParameterValues RPC on the CPE, reading PacketsReceived and PacketsResponded.
8. Allow the CPE to respond with the GetParameterValuesResponse
9. Orderly terminate the session
10. Use the UDP client at source to send n UDP Echo packets to the CPE
11. ACS schedules a GetParameterValues RPC on the CPE, reading PacketsReceived and PacketsResponded.
12. Allow the CPE to respond with the GetParameterValuesResponse

***Test Metrics:***
1. CPE responds correctly to the various RPCs
2. During step 4, the UPD client receives no response packet
3. During step 7, the counters read have a value of zero
4. During step 10, the UPD client receives n response packets (no loss expected in the closed test network)
5. During step 12, PacketsReceived and PacketsResponded are both n.

## 5.3 Statistics and Monitoring

### 5.3.1 Current Interface Configuration (Device:2 Only)

***Purpose:***

This test is designed to verify that the ACS can determine the current configuration of the interfaces on the CPE.

***References:***

Section 4/TR-181i2

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Baseline:2, IPInterface:1, EtherenetInterface:1, WiFiSSID:1, WiFiRadio:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

| **Device.InterfaceStack.**<br>**Device.IP.Interface.{i}.** | |
|---|---|
| **Enable** | Returned from device |
| **LowerLayers** | Returned from device |
| **Device.IP.Interface.{i}.IPv4Address.{i}.** | |
| **IPAddress** | Returned from device |
| **SubnetMask** | Returned from device |
| **Device.Ethernet.Interface.{i}.** | |
| **Enable** | Returned from device |
| **MACAddress** | Returned from device |
| **Device.Ethernet.Link.{i}.** | |
| **Enable** | Returned from device |
| **MACAddress** | Returned from device |
| **LowerLayers** | Returned from device |
| **Device.WiFi.SSID.{i}.** | |
| **Enable** | Returned from device |
| **MACAddress** | Returned from device |
| **LowerLayers** | Returned from device |
| **Device.WiFi.Radio.{i}.** | |
| **Enable** | Returned from device |
| **Name** | Returned from device |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

***Procedure:***

1. On the ACS schedule a GetParameterValues RPC for the InterfaceStack on the CPE.
2. Allow the CPE to respond with a GetParameterValuesResponse
3. On the ACS schedule a GetParameterValues on each interface returned in the InterfaceStack.

***Test Metrics:***

1. Verify that the CPE responds to the GetParameterValues RPC.
2. Verify that the InterfaceStack table refers to objects that exist.
3. Verify that each entry in the InterfaceStack table corresponds to the correct LowerLayer.
4. Verify Enable, IPAddress and SubnetMask or MACAddress from all Interface objects.

### 5.3.2 Connected LAN Devices (InternetGatewayDevice:1 Only)

***Purpose:***

This test is designed to verify that the ACS can determine the current connected LAN devices on the CPE and get back the correct response

***References:***

InternetGatewayDevice:1.4

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| N/A | Baseline:2 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

| InternetGatewayDevice.LANDevice.{i}.Hosts.Host.{i}. | |
|---|---|
| **IPAddress** | Returned from device |
| **AddressSource** | Returned from device |
| **LeaseTimeRemaining** | Returned from device |
| **MACAddress** | Returned from device |
| **Layer2Interface** | Returned from device |
| **HostName** | Returned from device |
| **InterfaceType** | Returned from device |
| **Active** | Returned from device |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have one LAN device connected on each interface that the CPE supports.
3. Have a Network Analyzer to capture traffic between the CPE and each LAN device.

***Procedure:***

1. Connect each LAN device so it is configured with an address.
2. On the ACS schedule a GetParameterValues RPC for InternetGatewayDevice.LANDevice. on the CPE.
3. Allow the CPE to respond with the GetParameterValues

***Test Metrics:***

1. Verify that the CPE responds to the GetParameterValues RPC
2. Verify that the Layer2Interface parameter matches the InterfaceType parameter.

3. Verify that the values of the above parameters returned match each LAN device connected to the CPE.
4. Verify that the Active parameter is set to “true”.

### 5.3.3 Connected LAN Devices – Wi-Fi (Device:2 Only)

***Purpose:***
This test is designed to verify that the ACS can determine the current connected LAN devices on the CPE and get back the correct response

***References:***
Device:2.2

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Hosts:2 | N/A |

***Optional Features***
None

***Parameters:***
The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.Hosts.Host.{i}.** | |
|---|---|
| **PhysAddress** | Returned from device |
| **AssociatedDevice** | Returned from device |
| **Layer1Interface** | Returned from device |
| **Layer3Interface** | Returned from device |
| **HostName** | Returned from device |
| **Active** | Returned from device |
| **Device.Hosts.Host.{i}.IPv4Address.{i}.** | |
| **IPAddress** | Returned from device |
| **Device.Hosts.Host.{i}.IPv6Address.{i}.** | |
| **IPAddress** | Returned from device |
| **Device.WiFi.AccessPoint.{i}.AssociatedDevice.{i}.** | |
| **MACAddress** | Returned from device |
| **SignalStrength** | Returned from device |
| **Retransmissions** | Returned from device |

Other AssociatedDevice tables if implemented

***Test Setup:***
1. Refer to Common Test Setup for setup steps.
2. Have one LAN device connected on the Wi-Fi interface of the CPE.
3. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***
1. Connect the LAN device so it is configured with an address.

2. On the ACS schedule a GetParameterValues RPC for Device.Hosts. on the CPE.
3. Allow the CPE to respond
4. On the ACS schedule a GetParameterValues RPC for the AssociatedDevice returned.
5. Allow the CPE to respond

***Test Metrics:***

1. Verify that the CPE responds to the GetParameterValues RPC
2. Verify that the values of the above parameters returned match the LAN device
3. Verify that the Active parameter is set to “true”
4. Verify that the value of SignalStrength is within the valid range
5. Verify that the value of Retransmissions is within the valid range

### 5.3.4 Connected LAN Devices – DHCP (Device:2 Only)

***Purpose:***

This test is designed to verify that the ACS can determine the current connected LAN devices on the CPE and get back the correct response

***References:***

Device:2.2

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Hosts:2, DHCPv4ServerClientInfo:1, DHCPv6ServerClientInfo:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.Hosts.Host.{i}.** | |
|---|---|
| **PhysAddress** | Returned from device |
| **DHCPClient** | Returned from device |
| **AssociatedDevice** | Returned from device |
| **Layer1Interface** | Returned from device |
| **Layer3Interface** | Returned from device |
| **HostName** | Returned from device |
| **Active** | Returned from device |
| **Device.Hosts.Host.{i}.IPv4Address.{i}.** | |
| **IPAddress** | Returned from device |
| **Device.Hosts.Host.{i}.IPv6Address.{i}.** | |
| **IPAddress** | Returned from device |
| **Device.WiFi.AccessPoint.{i}.AssociatedDevice.{i}.** | |
| **SignalStrength** | Returned from device |
| **Retransmissions** | Returned from device |
| **Device.DHCPv4.Server.Pool.{i}.Client.{i}.IPv4Address.{i}.** | |
| **LeaseTimeRemaining** | Returned from device |
| **Device.DHCPv4.Server.Pool.{i}.Client.{i}.Option.{i}.** | |
| **Tag** | Returned from device |
| **Value** | Returned from device |
| **Device.DHCPv6.Server.Pool.{i}.Client.{i}.IPv6Address.{i}.** | |
| **ValidLifetime** | Returned from device |

Other AssociatedDevice tables if implemented

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have one LAN device connected to the CPE.
3. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. Connect each LAN device so it is configured to use DHCP for its addressing information
2. On the ACS schedule a GetParameterValues RPC for Device.Hosts. on the CPE
3. Allow the CPE to respond
4. On the ACS schedule a GetParameterValues RPC for all the AssociatedDevices returned
5. Allow the CPE to respond
6. On the ACS schedule a GetParameterValues RPC for all the DHCPClient values returned

***Test Metrics:***

1. Verify that the CPE responds to the GetParameterValues RPC
2. Verify that the values of the above parameters returned match the LAN device
3. Verify that Option Tag 61 is the DHCP client id
4. Verify that the Active parameter is set to “true”

### 5.3.5 Device Connect/Disconnect Notification Test

***Purpose:***

This test is designed to verify that the ACS can determine when the set of LAN devices on the CPE changes and get back the correct response

***References:***

Device:2.0
InternetGatewayDevice:1.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| Hosts:1 | Baseline:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.Hosts.** | |
|---|---|
| **HostNumberOfEntries** | Returned from device |
| **Device.Hosts.Hosts.{i}.** | |
| **Active** | Returned from device |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.LANDevice.{i}.Hosts.** | |
|---|---|
| **HostNumberOfEntries** | Returned from device |
| **InternetGatewayDevice.LANDevice.{i}.Hosts.Host.{i}.** | |
| **Active** | Returned from device |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have one device connectable on each interface that the CPE supports.
3. Have a Network Analyzer to capture traffic between the CPE and each LAN device.

***Procedure:***

1. On the ACS schedule a SetParameterAttributes RPC with active notification on the CPE for HostNumberOfEntries
2. Connect a device to the CPE on each interface the CPE supports.
3. Verify that the ACS receives a notification containing the HostNumberOfEntries parameter.

4. On the ACS schedule a SetParameterAttributes RPC with active notification on the CPE for each host object's active parameter (Host.{i}.Active).
5. Disconnect a device from the CPE
6. Verify that the ACS receives a notification containing the Active parameter.
7. Schedule a GetParameterNames RPC on the CPE for Host.

***Test Metrics:***

1. Verify that the CPE responds to the SetParameterAttributes RPC
2. Verify the CPE sends an Inform containing a "4 VALUE CHANGE" for each device that is connected to the CPE. Verify the ParameterList includes the HostNumberOfEntries parameter.
3. Verify the CPE sends an Inform containing a "4 VALUE CHANGE" when the device is disconnected from the CPE. Verify the ParameterList includes the Host.{i}.Active parameter. If the CPE does not list inactive host, verify that the GetParameterNames does not include the Host instance of the inactive device.

## 5.4 Port Mappings

### 5.4.1 Create a Port Mapping – Single Interface

***Purpose:***

To verify that an ACS can create a port mapping, the CPE can create the port mapping that the ACS requested, and that internet traffic can traverse the port mapping.

***References:***

InternetGatewayDevice:1.0
Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| NAT:1 | Baseline:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented by the CPE for this test:

For devices that support the Device:2 root data model:

| **Device.NAT.PortMapping.** | |
|---|---|
| **Enable** | true |
| **Interface** | Returned from device |
| **LeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **Protocol** | UDP |
| **InternalClient** | <IP Address of the End Device (laptop)> |

For devices that support the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANIPConnection.{i}.PortMapping.{i}. OR InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.{i}.PortMapping.{i}.** | |
|---|---|
| **PortMappingEnabled** | true |
| **PortMappingLeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **PortMappingProtocol** | UDP |
| **InternalClient** | <IP Address of the End Device (laptop)> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

2. Have a LAN device connected to the LAN side of the CPE
3. Have a device connected on the WAN side of the CPE that can send UDP traffic.
4. ACS ensures that the port mapping being created doesn't already exist by retrieving all instances of the PortMapping table via a GetParamterValues RPC
    a. If PortMapping already exists, remove it via the DeleteObject RPC
5. Refer to Determine WAN Interface to determine the WAN interface.
6. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. ACS performs an AddObject RPC to create a new PortMapping instance.
2. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number.
3. ACS performs a GetParameterValues RPC on the PortMapping instance.
4. Reboot the CPE.
5. ACS performs a GetParameterValues RPC on the PortMapping instance.
6. Send UDP traffic to the CPE's IP Address with destination port 1400 from the WAN side of the CPE.

***Test Metrics:***

1. Validate that the values of the PortMapping instance match the values that were set.
2. Validate that the CPE retains the PortMapping instance and configuration across a reboot.
3. Validate that the UDP traffic is forwarded to the LAN Device's IP address, port 1401.

### 5.4.2 Create a Port Mapping – All Interfaces (Device:2 only)

***Purpose:***

To verify that an ACS can create a port mapping, the CPE can create the port mapping that the ACS requested, and that internet traffic can traverse the port mapping.

***References:***

Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| NAT:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented by the CPE for this test:

| **Device.NAT.PortMapping.** | |
|---|---|
| **Enable** | true |
| **AllInterfaces** | true |
| **LeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **Protocol** | UDP |
| **InternalClient** | <IP Address of the End Device (laptop)> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
   Have a LAN device connected to the LAN side of the CPE
2. Have a device connected on the WAN side of the CPE that can send UDP traffic.
3. ACS ensures that the port mapping being created doesn't already exist by retrieving all instances of the PortMapping table via a GetParamterValues RPC
   a. If PortMapping already exists, remove it via the DeleteObject RPC
4. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. ACS performs an AddObject RPC to create a new PortMapping instance.
2. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number.
3. ACS performs a GetParameterValues RPC on the PortMapping instance.
4. Send UDP traffic to the CPE's IP Address with destination port 1400 from the WAN side of the CPE.

***Test Metrics:***

1. Validate that the values of the PortMapping instance match the values that were set.
2. Validate that the UDP traffic is forwarded to the LAN Device's IP address, port 1401.

### 5.4.3 Create a Port Mapping – External Port Range

***Purpose:***

To verify that an ACS can create a port mapping, the CPE can create the port mapping that the ACS requested, and that internet traffic can traverse the port mapping.

***References:***

InternetGatewayDevice:1.4
and Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| NAT:1 | Baseline:1 |

***Optional Features***

ExternalPortEndRange is included in Data Model

***Parameters:***

The following parameters required to be implemented for this test:

For devices that support the Device:2 root data model:

| **Device.NAT.PortMapping.** | |
|---|---|
| **Enable** | true |
| **Interface** | Returned from device |
| **LeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **ExternalPortEndRange** | 1405 |
| **InternalPort** | 1401 |
| **Protocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

For devices that support the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANIPConnection.{i}.PortMapping.{i}. OR InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.{i}.PortMapping.{i}.** | |
|---|---|
| **PortMappingEnabled** | true |
| **PortMappingLeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **ExternalPortEndRange** | 1405 |
| **InternalPort** | 1401 |
| **PortMappingProtocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a LAN device connected to the LAN side of the CPE
3. Have a device connected on the WAN side of the CPE that can send UDP traffic.
4. ACS ensures that the port mapping being created doesn't already exist by retrieving all instances of the PortMapping table via a GetParamterValues RPC
    a. If PortMapping already exists, remove it via the DeleteObject RPC.
5. Refer to Determine WAN Interface to determine the WAN interface.
6. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. ACS performs an AddObject RPC to create a new PortMapping instance.
2. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number.
3. ACS performs a GetParameterValues RPC on the PortMapping instance.
4. Send UDP traffic to the CPE's IP Address with destination port 1400 from the WAN side of the CPE.
5. Send UDP traffic to the CPE's IP Address with destination port 1403 from the WAN side of the CPE.

***Test Metrics:***

1. Validate that the values of the PortMapping instance match the values that were set.
2. Validate that the UDP traffic to port 1400 is forwarded to the LAN Device's IP address, port 1401.
3. Validate that the UDP traffic to port 1403 is forwarded to the LAN Device's IP address, port 1401.

### 5.4.4 Create a Port Mapping - Lease Duration > 0

***Purpose:***

To verify that an ACS can create a port mapping, the CPE can create the port mapping that the ACS requested, and that internet traffic can traverse the port mapping.

***References:***

InternetGatewayDevice:1.0
Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| NAT:1 | Baseline:1 |

***Optional Features***

LeaseDuration or PoerMappingLeaseDuration supports non-zero value

***Parameters:***

The following parameters required to be implemented for this test:

For devices that support the Device:2 root data model:

| **Device.NAT.PortMapping.** | |
|---|---|
| **Enable** | true |
| **Interface** | Returned from device |
| **LeaseDuration** | 120 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **Protocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

For devices that support the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANIPConnection.{i}.PortMapping.{i}. OR InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.{i}.PortMapping.{i}.** | |
|---|---|
| **PortMappingEnabled** | true |
| **PortMappingLeaseDuration** | 120 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **PortMappingProtocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a LAN device connected to the LAN side of the CPE
3. Have a device connected on the WAN side of the CPE that can send UDP traffic.

4. ACS ensures that the port mapping being created doesn't already exist by retrieving all instances of the PortMapping table via a GetParamterValues RPC
    a. If PortMapping already exists, remove it via the DeleteObject RPC.
5. Refer to Determine WAN Interface to determine the WAN interface.
6. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. ACS performs an AddObject RPC to create a new PortMapping instance.
2. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number.
3. ACS performs a GetParameterValues RPC on the PortMapping instance.
4. Send UDP traffic to the CPE's IP Address with destination port 1400 from the WAN side of the CPE.
5. Wait for the PortMapping Instance to expire.
6. Send UDP traffic to the CPE's IP Address with destination port 1400 from the WAN side of the CPE.

***Test Metrics:***

1. Validate that the values of the PortMapping instance match the values that were set.
2. Validate that the UDP traffic to port 1400 is forwarded to the LAN Device's IP address, port 1401.
3. Validate that the UDP traffic to port 1400 is NOT forwarded to the LAN Device after the PortMapping Instance expires.

### *5.4.5* Create a Port Mapping – Remote Host Restriction

***Purpose:***

To verify that an ACS can create a port mapping, the CPE can create the port mapping that the ACS requested, and that internet traffic can traverse the port mapping.

***References:***

InternetGatewayDevice:1.0
and Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| NAT:1 | Baseline:1 |

***Optional Features***

RemoteHost supports a non-empty string

***Parameters:***

The following parameters required to be implemented for this test:

For devices that support the Device:2 root data model:

| **Device.NAT.PortMapping.** | |
|---|---|
| **Enable** | true |
| **Interface** | Returned from device |
| **LeaseDuration** | 0 |
| **RemoteHost** | <IP Address of Telnet Client1> |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **Protocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

For devices that support the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANIPConnection.{i}.PortMapping.{i}. OR InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.{i}.PortMapping.{i}.** | |
|---|---|
| **PortMappingEnabled** | true |
| **PortMappingLeaseDuration** | 0 |
| **RemoteHost** | <IP Address of one WANDevice1> |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **PortMappingProtocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a LAN device connected to the LAN side of the CPE

3. Have devices connected on the WAN side of the CPE: WANDevice1, and WANDevice2 that can both send UDP traffic.
4. ACS ensures that the port mapping being created doesn't already exist by retrieving all instances of the PortMapping table via a GetParamterValues RPC
    a. If PortMapping already exists, remove it via the DeleteObject RPC.
5. Refer to Determine WAN Interface to determine the WAN interface.
6. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. ACS performs an AddObject RPC to create a new PortMapping instance.
2. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number.
3. ACS performs a GetParameterValues RPC on the PortMapping instance.
4. Send UDP traffic to the CPE's IP Address with destination port 1400 from WANDevice1.
5. Send UDP traffic to the CPE's IP Address with destination port 1400 from WANDevice2

***Test Metrics:***

1. Validate that the values of the PortMapping instance match the values that were set.
2. Validate that the UDP traffic to port 1400 from WANDevice1 is forwarded to the LAN Device's IP address, port 1401.
3. Validate that the UDP traffic to port 1400 from WANDevice2 is NOT forwarded to the LAN Device.

### 5.4.6 Create a Port Mapping – Multiple Entries – Precedence Rules

***Purpose:***

To verify that an ACS can create a port mapping, the CPE can create the port mapping that the ACS requested, and that internet traffic can traverse the port mapping.

***References:***

InternetGatewayDevice:1.0
and Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| NAT:1 | Baseline:1 |

***Optional Features***

None

***Parameters:***

The following parameters required to be implemented for this test:

For devices that support the Device:2 root data model:

| **Device.NAT.PortMapping.** | |
|---|---|
| **Enable** | true |
| **Interface** | Returned from device |
| **LeaseDuration** | 0 |
| **ExternalPort** | 0 |
| **InternalPort** | 1401 |
| **Protocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

For devices that support the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANIPConnection.{i}.PortMapping.{i}. OR InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.{i}.PortMapping.{i}.** | |
|---|---|
| **PortMappingEnabled** | true |
| **PortMappingLeaseDuration** | 0 |
| **ExternalPort** | 0 |
| **InternalPort** | 1401 |
| **PortMappingProtocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a LAN device connected to the LAN side of the CPE
3. Have two devices connected on the WAN side of the CPE: WANDevice1, and WANDevice2 that can both send UDP traffic.

4. ACS ensures that the port mapping being created doesn't already exist by retrieving all instances of the PortMapping table via a GetParamterValues RPC
    a. If PortMapping already exists, remove it via the DeleteObject RPC.
5. Refer to Determine WAN Interface to determine the WAN interface.
6. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. ACS performs an AddObject RPC to create a new PortMapping instance.
2. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number. ExternalPort is 0, InternalPort is 1401.
3. ACS performs an AddObject RPC to create a new PortMapping instance.
4. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number. ExternalPort is 1400, InternalPort is 1402.
5. ACS performs a GetParameterValues RPC on the PortMapping instance.
6. Send UDP traffic to the CPE's IP Address with destination port 1399 from WANDevice1.
7. Send UDP traffic to the CPE's IP Address with destination port 1400 from WANDevice2.

***Test Metrics:***

1. Validate that the values of the PortMapping instance match the values that were set.
2. Validate that the UDP traffic to port 1399 from WANDevice1 is forwarded to the LAN Device's IP address, port 1401.
3. Validate that the UDP traffic to port 1400 from WANDevice2 is forwarded to the LAN Device's IP address, port 1402.

### 5.4.7 Modify a Port Mapping

***Purpose:***

To verify that an ACS can change a port mapping, the CPE can change the port mapping in the way that the ACS requested, and that internet traffic can traverse the altered port mapping.

***References:***

InternetGatewayDevice:1.0
Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| NAT:1 | Baseline:1 |

***Optional Features***

None

***Parameters:***

The following parameters required to be implemented for this test:

For devices that support the Device:2 root data model:

| **Device.NAT.PortMapping.** | |
|---|---|
| **Enable** | true |
| **Interface** | Returned from device |
| **LeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **Protocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

For devices that support the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANIPConnection.{i}.PortMapping.{i}. OR InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.{i}.PortMapping.{i}.** | |
|---|---|
| **PortMappingEnabled** | true |
| **PortMappingLeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **PortMappingProtocol** | UDP |
| **InternalClient** | <IP Address of the End Device> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a LAN device connected to the LAN side of the CPE
3. Have a device connected on the WAN side of the CPE that can send UDP traffic.

4. ACS ensures that the port mapping being created doesn't already exist by retrieving all instances of the PortMapping table via a GetParamterValues RPC
    a. If PortMapping already exists, remove it via the DeleteObject RPC.
5. Refer to Determine WAN Interface to determine the WAN interface.
6. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. ACS performs an AddObject RPC to create a new PortMapping instance.
2. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number. ExternalPort is 1400, InternalPort is 1401.
3. Send UDP traffic to the CPE's IP Address with destination port 1400 from the WAN side of the CPE.
4. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number. ExternalPort is 1401, InternalPort is 1402.
5. ACS performs a GetParameterValues RPC on the PortMapping instance.
6. Send UDP traffic to the CPE's IP Address with destination port 1401 from the WAN side of the CPE.

***Test Metrics:***

1. Validate that the UDP traffic to port 1400 is forwarded to the LAN Device's IP address, port 1401.
2. Validate that the values of the PortMapping instance match the values that were set.
3. Validate that the UDP traffic to port 1401 is forwarded to the LAN Device's IP address, port 1402.

### 5.4.8 Delete a Port Mapping

***Purpose:***

To verify that an ACS can remove a port mapping, the CPE can remove the port mapping that the ACS requested, and that internet traffic will not traverse the across the removed port mapping.

***References:***

InternetGatewayDevice:1.0
and Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| NAT:1 | Baseline:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented by the CPE for this test:

For devices that support the Device:2 root data model:

| **Device.NAT.PortMapping.** | |
|---|---|
| **Enable** | true |
| **Interface** | Returned from device |
| **LeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **Protocol** | UDP |
| **InternalClient** | <IP Address of the End Device (laptop)> |

For devices that support the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANIPConnection.{i}.PortMapping.{i}.<br>OR<br>InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.{i}.PortMapping.{i}.** | |
|---|---|
| **PortMappingEnabled** | true |
| **PortMappingLeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **PortMappingProtocol** | UDP |
| **InternalClient** | <IP Address of the End Device (laptop)> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a LAN device connected to the LAN side of the CPE
3. Have a device connected on the WAN side of the CPE that can send UDP traffic.

4. ACS ensures that the port mapping being altered already exists by retrieving all instances of the PortMapping table via a GetParamterValues RPC
    a. If PortMapping does NOT already exist, add it via the DeleteObject/SetParameterValues RPC: see “Create a PortMapping (single interface)” test for procedures
5. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. Send UDP traffic to the CPE’s IP Address with destination port 1400 from the WAN side of the CPE.
2. ACS performs a DeleteObject RPC on the PortMapping Instance.
3. ACS validates that the PortMapping entry no longer exists by retrieving all instances of the PortMapping table via a GetParamterValues RPC
4. Send UDP traffic to the CPE’s IP Address with destination port 1400 from the WAN side of the CPE.

***Test Metrics:***

1. Validate that the UDP traffic to port 1400 is forwarded to the LAN Device’s IP address, port 1401.
2. Validate that the PortMapping Instance no longer exists in the table.
3. Validate that the UDP traffic to port 1400 is NOT forwarded to the LAN Device after the PortMapping has been deleted.

### 5.4.9 Create a Port Mapping – TCP

***Purpose:***

To verify that an ACS can create a port mapping, the CPE can create the port mapping that the ACS requested, and that internet traffic can traverse the port mapping.

***References:***

InternetGatewayDevice:1.0
Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| NAT:1 | Baseline:1 |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented by the CPE for this test:

For devices that support the Device:2 root data model:

| Device.NAT.PortMapping. | |
|---|---|
| **Enable** | true |
| **Interface** | Returned from device |
| **LeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **Protocol** | TCP |
| **InternalClient** | <IP Address of the End Device (laptop)> |

For devices that support the InternetGatewayDevice:1 root data model:

| InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANIPConnection.{i}.PortMapping.{i}. OR InternetGatewayDevice.WANDevice.{i}.WANConnectionDevice.{i}.WANPPPConnection.{i}.PortMapping.{i}. | |
|---|---|
| **PortMappingEnabled** | true |
| **PortMappingLeaseDuration** | 0 |
| **ExternalPort** | 1400 |
| **InternalPort** | 1401 |
| **PortMappingProtocol** | TCP |
| **InternalClient** | <IP Address of the End Device (laptop)> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a LAN device connected to the LAN side of the CPE
3. Have a device connected on the WAN side of the CPE that can send TCP traffic.

4. ACS ensures that the port mapping being created doesn't already exist by retrieving all instances of the PortMapping table via a GetParamterValues RPC
    a. If PortMapping already exists, remove it via the DeleteObject RPC
5. Refer to Determine WAN Interface to determine the WAN interface.
6. Have a Network Analyzer to capture traffic between the CPE and the LAN device.

***Procedure:***

1. ACS performs an AddObject RPC to create a new PortMapping instance.
2. ACS performs a SetParameterValues RPC on the parameters above using the returned PortMapping instance number.
3. ACS performs a GetParameterValues RPC on the PortMapping instance.
4. Reboot the CPE.
5. ACS performs a GetParameterValues RPC on the PortMapping instance.
6. Send TCP traffic to the CPE's IP Address with destination port 1400 from the WAN side of the CPE.

***Test Metrics:***

1. Validate that the values of the PortMapping instance match the values that were set.
2. Validate that the CPE retains the PortMapping instance and configuration across a reboot.
3. Validate that the TCP traffic is forwarded to the LAN Device's IP address, port 1401.

## 5.5 Advanced Firewall

### 5.5.1 Default Policy (Device:2 Only)

***Purpose:***

To verify the ACS can configure a firewall with a default policy on the CPE.

***References:***

Device:2.2

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| AdvancedFirewall:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

| **Device.Firewall.** | |
|---|---|
| **Enable** | true |
| **Config** | Advanced |
| **AdvancedLevel** | Device.Firewall.Level.{i}. |
| **Device.Firewall.Level.{i}.** | |
| **Chain** | Returned from device |
| **DefaultPolicy** | Accept |
| **Device.Firewall.Chain.{i}.** | |
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Drop |
| **SourceIP** | <Unused IP Address> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a device that can send and receive TCP and UDP connections connected to the LAN side of the CPE (End Device)
3. Have a device that can send and receive TCP and UDP connections connected on the WAN side of the CPE (WAN Device)
4. Have a Network Analyzer to capture traffic between the CPE and the End Device.

***Procedure:***

1. Configure ACS to send an AddObject RPC for Device.Firewall.Level and record the returned instance number.
2. Configure ACS to send a GetParameterValues RPC for Device.Firewall.Level.{i}.Chain.

3. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule using the path returned in step 2 and record the returned instance number.
4. ACS performs a SetParameterValues RPC on the parameters above except Device.Firewall.Enable using the returned instance numbers from steps 1-3.
5. ACS enables the Firewall by sending a SetParameterValues RPC for Device.Firewall.Enable with the value set to "true."
6. End Device attempts to open a TCP connection with WAN Device.
7. WAN Device attempts to open a TCP connection with End Device.
8. ACS configures the Firewall by sending a SetParameterValues RPC with following Name/Value pairs:

| **Device.Firewall.Level.{i}.** | |
|---|---|
| **DefaultPolicy** | Drop |

9. End Device attempts to open a TCP connection WAN Device.
10. WAN Device attempts to open a TCP connection with End Device.

***Test Metrics:***
1. In Step 6, CPE forwards traffic to the WAN side.
2. In Step 7, CPE forwards traffic to the LAN side.
3. In Step 9, CPE does not forward traffic to the WAN side.
4. In Step 10, CPE does not forward traffic to the LAN side.

### 5.5.2 Deny/Allow Outbound Protocols (Device:2 Only)

***Purpose:***

To verify the ACS can configure a firewall on the CPE that denies or allows specific outbound protocols.

***References:***

Device:2.2

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| AdvancedFirewall:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

| **Device.Firewall.** | |
|---|---|
| **Enable** | true |
| **Config** | Advanced |
| **AdvancedLevel** | Device.Firewall.Level.{i}. |
| **Device.Firewall.Level.{i}.** | |
| **Chain** | Returned from device |
| **DefaultPolicy** | Drop |
| **Device.Firewall.Chain.{i}.** | |
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **SourceIP** | <Unused IP Address> |
| **DestInterface** | Default WAN Interface |
| **Protocol** | 6 (TCP), 17 (UDP) |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Refer to Determine WAN Interface to determine the WAN interface.
3. Device that can send and receive TCP and UDP connections connected to the LAN side of the CPE (End Device)
4. Device that can send and receive TCP and UDP connections connected on the WAN side of the CPE (WAN Device)
5. Have a Network Analyzer to capture traffic between the CPE and the End Device.

***Procedure:***

1. Configure ACS to send an AddObject RPC for Device.Firewall.Level and record the returned instance number.
2. Configure ACS to send a GetParameterValues RPC for Device.Firewall.Level.{i}.Chain.
3. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule using the path returned in step 2 and record the returned instance number.
4. ACS performs a SetParameterValues RPC on the following parameters using the instance numbers returned in steps 1-3.

| **Device.Firewall.** | |
|---|---|
| **Config** | Advanced |
| **AdvancedLevel** | Device.Firewall.Level.{i}. |
| **Device.Firewall.Level.{i}.** | |
| **Chain** | Returned from device |
| **DefaultPolicy** | Drop |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestIP** | <Unused IP Address> |
| **SourceIP** | <Unused IP Address> |

5. ACS enables the Firewall by sending a SetParameterValues RPC for Device.Firewall.Enable with the value set to “true.”
6. End Device attempts to open a TCP connection with WAN Device.
7. End Device attempts to open a UDP connection with WAN Device.
8. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule. and record the returned instance number.
9. ACS Configures the new Firewall Rule by sending a SetParameterValues RPC with the following name/value pairs using the instance number returned in step 8.

| **Device.Firewall.Chain.{i}.** | |
|---|---|
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **Protocol** | 6 (TCP) |

10. End Device attempts to open a TCP connection with WAN Device.
11. End Device attempts to open a UDP connection with WAN Device.
12. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule. and record the returned instance number.
13. ACS Configures the new Firewall Rule by sending a SetParameterValues RPC with the following name/value pairs using the instance number returned in step 12.

| **Device.Firewall.Chain.{i}.** | |
|---|---|
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |

| | |
|---|---|
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **Protocol** | 17 (UDP) |

14. End Device attempts to open a TCP connection with WAN Device.
15. End Device attempts to open a UDP connection with WAN Device.

***Test Metrics:***

1. In Step 6, CPE does not forward TCP traffic to the WAN side.
2. In Step 7, CPE does not forward UDP traffic to the WAN side.
3. In Step 10, CPE forwards TCP traffic to the WAN side.
4. In Step 11, CPE does not forward UDP traffic to the WAN side.
5. In Step 14, CPE forwards TCP traffic to the WAN side.
6. In Step 15, CPE forwards UDP traffic to the WAN side.

### 5.5.3 Deny/Allow Outbound Ports (Device:2 Only)

***Purpose:***

To verify the ACS can configure a firewall on the CPE that denies or allows specific outbound ports.

***References:***

Device:2.2

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| AdvancedFirewall:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

| **Device.Firewall.** | |
|---|---|
| **Enable** | true |
| **Config** | Advanced |
| **AdvancedLevel** | Device.Firewall.Level.{i}. |
| **Device.Firewall.Level.{i}.** | |
| **Chain** | Returned from device |
| **DefaultPolicy** | Drop |
| **Device.Firewall.Chain.{i}.** | |
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **DestIP** | <Unused IP Address> |
| **SourceIP** | <Unused IP Address> |
| **Protocol** | 6 (TCP), 17 (UDP) |
| **DestPort** | int[-1:65535] |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Refer to Determine WAN Interface to determine the WAN interface.
3. Device that can send and receive TCP and UDP connections connected to the LAN side of the CPE (End Device)
4. Device that can send and receive TCP and UDP connections connected on the WAN side of the CPE (WAN Device)
5. Have a Network Analyzer to capture traffic between the CPE and the End Device.

***Procedure:***

1. Configure ACS to send an AddObject RPC for Device.Firewall.Level and record the returned instance number.
2. Configure ACS to send a GetParameterValues RPC for Device.Firewall.Level.{i}.Chain.
3. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule using the path returned in step 2 and record the returned instance number.
4. ACS performs a SetParameterValues RPC on the following parameters using the instance numbers returned in steps 1-3.

| **Device.Firewall.** | |
|---|---|
| **Config** | Advanced |
| **AdvancedLevel** | Device.Firewall.Level.{i}. |
| **Device.Firewall.Level.{i}.** | |
| **Chain** | Returned from device |
| **DefaultPolicy** | Drop |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestIP** | <Unused IP Address> |
| **SourceIP** | <Unused IP Address> |

5. ACS enables the Firewall by sending a SetParameterValues RPC for Device.Firewall.Enable with the value set to “true.”
6. End Device attempts to open a TCP connection, destination port 80, with WAN Device.
7. End Device attempts to open a TCP connection, destination port 443, with WAN Device.
8. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule and record the returned instance number.
9. ACS Configures the new Firewall Rule by sending a SetParameterValues RPC with the following name/value pairs using the instance number returned in step 8.

| **Device.Firewall.Chain.{i}.** | |
|---|---|
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **Protocol** | 6 |
| **DestPort** | 80 |

10. End Device attempts to open a TCP connection, destination port 80, with WAN Device.
11. End Device attempts to open a TCP connection, destination port 443, with WAN Device.
12. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule and record the returned instance number.
13. ACS Configures the new Firewall Rule by sending a SetParameterValues RPC with the following name/value pairs using the instance number returned in step 12.

| **Device.Firewall.Chain.{i}.** | |
|---|---|
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |

| | |
|---|---|
| **DestInterface** | Default WAN Interface |
| **Protocol** | 6 |
| **DestPort** | 443 |

14. End Device attempts to open a TCP connection, destination port 80, with WAN Device.
15. End Device attempts to open a TCP connection, destination port 443, with WAN Device.

***Test Metrics:***

1. In Step 6, CPE does not forward TCP traffic with destination port 80 to the WAN side.
2. In Step 7, CPE does not forward TCP traffic with destination port 443 to the WAN side.
3. In Step 10, CPE forwards TCP traffic with destination port 80 to the WAN side.
4. In Step 11, CPE does not forward TCP traffic with destination port 443to the WAN side.
5. In Step 14, CPE forwards TCP traffic with destination port 80 to the WAN side.
6. In Step 15, CPE forwards TCP traffic with destination port 443 to the WAN side.

### 5.5.4 Deny/Allow Source IP Address (Device:2 Only)

***Purpose:***
To verify the ACS can configure a firewall on the CPE that denies or allows specific outbound ports.

***References:***
Device:2.2

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| AdvancedFirewall:1 | N/A |

***Optional Features***
None

***Parameters:***
The following parameters are required to be implemented for this test:

| **Device.Firewall.** | |
|---|---|
| **Enable** | true |
| **Config** | Advanced |
| **AdvancedLevel** | Device.Firewall.Level.{i}. |
| **Device.Firewall.Level.{i}.** | |
| **Chain** | Returned from device |
| **DefaultPolicy** | Drop |
| **Device.Firewall.Chain.{i}.** | |
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **DestIP** | <Unused IP Address> |
| **SourceIP** | <Unused IP Address> |
| **Protocol** | 6 (TCP), 17 (UDP) |
| **DestPort** | int[-1:65535] |

***Test Setup:***
1. Refer to Common Test Setup for setup steps.
2. Refer to Determine WAN Interface to determine the WAN interface.
3. Two Devices that can send and receive TCP and UDP connections connected to the LAN side of the CPE (End Device 1 and End Device 2)
4. Device that can send and receive TCP and UDP connections connected on the WAN side of the CPE (WAN Device)
5. Have a Network Analyzer to capture traffic between the CPE and the End Device.

***Procedure:***

1. Configure ACS to send an AddObject RPC for Device.Firewall.Level and record the returned instance number.
2. Configure ACS to send a GetParameterValues RPC for Device.Firewall.Level.{i}.Chain.
3. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule using the path returned in step 2 and record the returned instance number.
4. ACS performs a SetParameterValues RPC on the following parameters using the instance numbers returned in steps 1-3.

| **Device.Firewall.** | |
|---|---|
| **Config** | Advanced |
| **AdvancedLevel** | Device.Firewall.Level.{i}. |
| **Device.Firewall.Level.{i}.** | |
| **Chain** | Returned from device |
| **DefaultPolicy** | Drop |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestIP** | <Unused IP Address> |
| **SourceIP** | <Unused IP Address> |

5. ACS enables the Firewall by sending a SetParameterValues RPC for Device.Firewall.Enable with the value set to “true.”
6. End Device 1 attempts to open a TCP connection, destination port 80, with WAN Device.
7. End Device 2 attempts to open a TCP connection, destination port 80, with WAN Device.
8. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule and record the returned instance number.
9. ACS Configures the new Firewall Rule by sending a SetParameterValues RPC with the following name/value pairs using the instance number returned in step 8.

| **Device.Firewall.Chain.{i}.** | |
|---|---|
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **Protocol** | 6 |
| **DestPort** | 80 |
| **SourceIP** | End Device1 IP |

10. End Device 1 attempts to open a TCP connection, destination port 80, with WAN Device.
11. End Device 2 attempts to open a TCP connection, destination port 80, with WAN Device.
12. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule and record the returned instance number.
13. ACS Configures the new Firewall Rule by sending a SetParameterValues RPC with the following name/value pairs using the instance number returned in step 12.

| **Device.Firewall.Chain.{i}.** | |
|---|---|
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |

| | |
|---|---|
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **Protocol** | 6 |
| **DestPort** | 80 |
| **SourceIP** | End Device2 IP |

14. End Device 1 attempts to open a TCP connection, destination port 80, with WAN Device.
15. End Device 2 attempts to open a TCP connection, destination port 80 with WAN Device.

***Test Metrics:***

1. In Step 6, CPE does not forward TCP traffic from End Device 1 to the WAN side.
2. In Step 7, CPE does not forward TCP traffic from End Device 2 to the WAN side.
3. In Step 10, CPE forwards TCP traffic from End Device 1 to the WAN side.
4. In Step 11, CPE does not forward TCP traffic from End Device 2 to the WAN side.
5. In Step 14, CPE forwards TCP traffic from End Device 1 to the WAN side.
6. In Step 15, CPE forwards TCP traffic from End Device 2 to the WAN side.

## 5.6 Wi-Fi Provisioning

### 5.6.1 Wi-Fi Setup WEP 64 (InternetGatewayDevice:1 Only)

***Purpose:***

To verify that an ACS can set a valid wireless LAN configuration on the CPE using WEP 64 encryption.

***References:***

InternetGatewayDevice:1.4

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| N/A | WiFiLan:1 |

***Optional Features***

WEPEncryptionLevel includes "40-bit"

***Parameters:***

The following parameters are required to be implemented for this test. A CPE must support WEP-64 encryption in order to run this test.

| **InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.** | |
|---|---|
| **Enable** | true |
| **RadioEnabled** | true |
| **SSID** | <string> |
| **Channel** | <unsignedint> |
| **BeaconType** | Basic |
| **BeaconAdvertisementEnabled** | true |
| **BasicEncryptionModes** | WEPEncryption |
| **WEPEncryptionLevel** | Returned from device |
| **BasicAuthenticationMode** | None |
| **InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.WEPKey.{i}.** | |
| **WEPKey** | 5-character hexadecimal string |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the WLANConfiguration partial path listed in the parameters section to learn the existing stack hierarchy.
2. Configure the wireless LAN settings on the CPE using a single SetParameterValues.
3. Reboot the CPE.

4. The End Device connects to the CPE’s newly created wireless LAN access point.
5. Stimulate network traffic across the wireless LAN connection

***Test Metrics:***
1. Verify that WEPEncryptionLevel includes “40-bit”.
2. Verify that the SetParameterValuesResponse is valid.
3. Settings persist across reboot.
4. The End Device can connect to the wireless LAN connection.
5. The End Device has appropriate network access through the wireless LAN connection.

### 5.6.2 Wi-Fi Setup WEP 128 (InternetGatewayDevice:1 Only)

***Purpose:***

To verify that an ACS can set a valid wireless LAN configuration on the CPE using WEP 128 encryption.

***References:***

InternetGatewayDevice:1.4

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| N/A | WiFiLan:1 |

***Optional Features***

WEPEncryptionLevel includes "104-bit"

***Parameters:***

The following parameters are required to be implemented for this test. A CPE must support WEP-128 encryption in order to run this test.

| **InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.** | |
|---|---|
| **Enable** | true |
| **RadioEnabled** | true |
| **SSID** | <string> |
| **Channel** | <unsignedint> |
| **BeaconType** | Basic |
| **BeaconAdvertisementEnabled** | true |
| **BasicEncryptionModes** | WEPEncryption |
| **WEPEncryptionLevel** | Returned from Device |
| **BasicAuthenticationMode** | None |
| **InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.WEPKey.{i}.** | |
| **WEPKey** | 13-character hexadecimal string |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the WLANConfiguration partial path listed in the parameters section to learn the existing stack hierarchy.
2. Configure the wireless LAN settings on the CPE using a single SetParameterValues.
3. Reboot the CPE.
4. The End Device connects to the CPE's newly created wireless LAN access point.
5. Stimulate network traffic across the wireless LAN connection

***Test Metrics:***

1. Verify that WEPEncryptionLevel includes “104-bit”.
2. Verify that the SetParameterValuesResponse is valid.
3. Settings persist across reboot.
4. The End Device can connect to the wireless LAN connection.
5. The End Device has appropriate network access through the wireless LAN connection.

### 5.6.3 Wi-Fi Setup WPA Personal (InternetGatewayDevice:1 Only)

***Purpose:***

To verify that an ACS can set a valid Wireless LAN configuration on the CPE using WPA Personal encryption.

***References:***

InternetGatewayDevice:1.4

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| N/A | WiFiLan:1 |

***Optional Features***

BeaconType supports "WPA"

***Parameters:***

The following parameters are required to be implemented for this test.

| **InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.** | |
|---|---|
| **Enable** | true |
| **RadioEnabled** | true |
| **SSID** | <string> |
| **Channel** | <unsignedint> |
| **BeaconType** | WPA |
| **BeaconAdvertisementEnabled** | true |
| **WPAEncryptionModes** | AESEncryption |
| **WPAAuthenticationMode** | PSKAuthentication |
| **InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.PreSharedKey.{i}.** | |
| **PreSharedKey** | Hexadecimal string |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the WLANConfiguration partial path listed in the parameters section to learn the existing stack hierarchy.
2. Configure the wireless LAN settings on the CPE using a single SetParameterValues.
3. Reboot the CPE.
4. The End Device connects to the CPE's newly created wireless LAN access point.
5. Stimulate network traffic across the wireless LAN connection

***Test Metrics:***

1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. The End Device can connect to the wifi connection.
4. The End Device has appropriate network access through the wifi connection.

### 5.6.4 Wi-Fi Setup WPA2 Personal (InternetGatewayDevice:1 Only)

***Purpose:***

To verify that an ACS can set a valid Wireless LAN configuration on the CPE using WPA2-Personal encryption.

***References:***

InternetGatewayDevice:1.4

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| N/A | WiFiLan:1 |

***Optional Features***

BeaconType supports "11i"

***Parameters:***

The following parameters are required to be implemented for this test.

| **InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.** | |
|---|---|
| **Enable** | true |
| **RadioEnabled** | true |
| **SSID** | <string> |
| **Channel** | <unsignedint> |
| **BeaconType** | 11i |
| **BeaconAdvertisementEnabled** | true |
| **IEEE11iEncryptionModes** | AESEncryption |
| **IEEE11iAuthenticationMode** | PSKAuthentication |
| **InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.PreSharedKey.{i}.** | |
| **PreSharedKey** | Hexadecimal string |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks

***Procedure:***

1. Perform a GetParameterValues on the WLANConfiguration partial path listed in the parameters section to learn the existing stack hierarchy.
2. Configure the wireless LAN settings on the CPE using a single SetParameterValues.
3. Reboot the CPE.
4. The End Device connects to the CPE's newly created wireless LAN access point.
5. Stimulate network traffic across the wireless LAN connection

***Test Metrics:***

1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. The End Device can connect to the wifi connection.
4. The End Device has appropriate network access through the wifi connection.

### 5.6.5 Wi-Fi Setup – WPA-WPA2-Personal (InternetGatewayDevice:1 Only)

***Purpose:***

To verify that an ACS can set a valid Wireless LAN configuration on the CPE using WPA-WPA2-Personal encryption.

***References:***

InternetGatewayDevice:1.4

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| N/A | WiFiLan:1 |

***Optional Features***

BeaconType includes "WPAand11i"

***Parameters:***

The following parameters are required to be implemented for this test.

| InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}. | |
|---|---|
| **Enable** | true |
| **RadioEnabled** | true |
| **SSID** | <string> |
| **Channel** | <unsignedint> |
| **BeaconType** | WPAand11i |
| **BeaconAdvertisementEnabled** | true |
| **IEEEEncryptionModes** | AESEncryption |
| **IEEEAuthenticationMode** | PSKAuthentication |
| **InternetGatewayDevice.LANDevice.{i}.WLANConfiguration.{i}.PreSharedKey.{i}.** | |
| **PreSharedKey** | Hexadecimal string |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the WLANConfiguration partial path listed in the parameters section to learn the existing stack hierarchy.
2. Configure the wireless LAN settings on the CPE using a single SetParameterValues.
3. Reboot the CPE.
4. The End Device connects to the CPE's newly created wireless LAN access point.
5. Stimulate network traffic across the wireless LAN connection

***Test Metrics:***

1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. The End Device can connect to the wifi connection.
4. The End Device has appropriate network access through the wifi connection.

### 5.6.6 Wi-Fi Setup WEP 64 (Device:2 Only)

***Purpose:***

To verify that an ACS can set a valid wireless LAN configuration on the CPE using WEP 64 encryption.

***References:***

Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

ModesSupported includes "WEP-64"

***Parameters:***

The following parameters are required to be implemented for this test. A CPE must support WEP-64 encryption in order to run this test.

| **Device.WiFi.Radio.{i}.** | |
|---|---|
| **Enable** | true |
| **Channel** | <unsignedInt> |
| **OperatingFrequencyBand** | 2.4GHz or 5GHz |
| **OperatingStandards** | <appropriate setting> |
| **Device.WiFi.SSID.{i}.** | |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDAdvertisementEnabled** | true |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.AccessPoint.{i}.Security.** | |
| **ModeEnabled** | WEP-64 |
| **ModesSupported** | Returned from device |
| **WEPKey** | 5-character hexadecimal string |
| **Device.WiFi.Radio.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy.
2. If no SSID objects exist, perform an AddObject RPC on the Device.Wifi.SSID. partial path.
    a. Record the instance number received in the AddObjectResponse.
3. If no AccessPoint objects exist, perform an AddObject RPC on the Device.Wifi.AccessPoint. partial path
    a. Record the instance number received in the AddObjectResponse
4. Configure the Wifi settings on the CPE using a single SetParameterValues.
5. Reboot the CPE.
6. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.
7. The End Device connects to the CPE's newly created wifi access point.
8. Stimulate network traffic across the Wifi connection
9. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.

***Test Metrics:***
1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. The End Device can connect to the wireless LAN connection.
4. The End Device has appropriate network access through the wireless LAN connection.
5. The Stats object has updated to reflect the network traffic across the wireless LAN connection.

### 5.6.7 Wi-Fi Setup WEP 128 (Device:2 Only)

***Purpose:***

To verify that an ACS can set a valid WiFi configuration on the CPE using WEP 128 encryption.

***References:***

Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

ModesSupported includes "WEP-128"

***Parameters :***

The following parameters are required to be implemented for this test. A CPE must support WEP-128 encryption in order to run this test.

| **Device.WiFi.Radio.{i}.** | |
|---|---|
| **Enable** | true |
| **Channel** | <unsignedInt> |
| **OperatingFrequencyBand** | 2.4GHz or 5GHz |
| **OperatingStandards** | Auto |
| **Device.WiFi.SSID.{i}.** | |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDAdvertisementEnabled** | true |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.AccessPoint.{i}.Security.** | |
| **ModeEnabled** | WEP-128 |
| **ModesSupported** | Returned from device |
| **WEPKey** | 13-character hexadecimal string |
| **Device.WiFi.Radio.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy.
2. If no SSID objects exist, perform an AddObject RPC on the Device.Wifi.SSID. partial path.
   a. Record the instance number received in the AddObjectResponse.
3. If no AccessPoint objects exist, perform an AddObject RPC on the Device.Wifi.AccessPoint. partial path
   a. Record the instance number received in the AddObjectResponse
4. Configure the Wifi settings on the CPE using a single SetParameterValues.
5. Reboot the CPE.
6. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.
7. The End Device connects to the CPE's newly created wifi access point.
8. Stimulate network traffic across the Wifi connection
9. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.

***Test Metrics:***
1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. The End Device can connect to the wifi connection.
4. The End Device has appropriate network access through the wifi connection.
5. The Stats object has updated to reflect the network traffic across the wifi connection.

### 5.6.8 Wi-Fi Setup WPA Personal (Device:2 Only)

***Purpose:***

To verify that an ACS can set a valid WiFi configuration on the CPE using WPA Personal Encryption.

***References:***

Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

ModesSupported includes "WPA-Personal"

***Parameters:***

The following parameters are required to be implemented for this test.

| **Device.WiFi.Radio.{i}.** | |
|---|---|
| **Enable** | true |
| **Channel** | <unsignedInt> |
| **OperatingFrequencyBand** | 2.4GHz or 5GHz |
| **OperatingStandards** | Auto |
| **Device.WiFi.SSID.{i}.** | |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDAdvertisementEnabled** | true |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.AccessPoint.{i}.Security.** | |
| **ModeEnabled** | WPA-Personal |
| **ModesSupported** | Returned from device |
| **PreSharedKey** | hexadecimal string |
| **Device.WiFi.Radio.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

3. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy.

4. If no SSID objects exist, perform an AddObject RPC on the Device.Wifi.SSID. partial path.
    a. Record the instance number received in the AddObjectResponse.
5. If no AccessPoint objects exist, perform an AddObject RPC on the Device.Wifi.AccessPoint. partial path
    a. Record the instance number received in the AddObjectResponse
6. Configure the Wifi settings on the CPE using a single SetParameterValues.
7. Reboot the CPE.
8. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.
9. The End Device connects to the CPE's newly created wifi access point.
10. Stimulate network traffic across the Wifi connection
11. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.

***Test Metrics:***
1. Verify that the SetParameterValuesResponse to be valid.
2. Settings persist across reboot.
3. The End Device can connect to the wifi connection.
4. The End Device has appropriate network access through the wifi connection.
5. The Stats object has updated to reflect the network traffic across the wifi connection.

### 5.6.9 Wi-Fi Setup WPA Enterprise (Device:2 Only)

***Purpose:***

To verify that an ACS can set a valid WiFi configuration on the CPE using WPA Enterprise Encryption.

***References:***

Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

ModesSupported includes "WPA-Enterprise"

***Parameters:***

The following parameters are required to be implemented for this test.

| **Device.WiFi.Radio.{i}.** | |
|---|---|
| **Enable** | true |
| **Channel** | <unsignedInt> |
| **OperatingFrequencyBand** | 2.4GHz or 5GHz |
| **OperatingStandards** | Auto |
| **Device.WiFi.SSID.{i}.** | |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDAdvertisementEnabled** | true |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.AccessPoint.{i}.Security.** | |
| **ModeEnabled** | WPA-Enterprise |
| **RadiusServerIPAddr** | <IP address of RADIUS server> |
| **RadiusServerPort** | <Port of RADIUS Server> |
| **RadiusSecret** | <string> |
| **Device.WiFi.Radio.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a RADIUS Server set up configured with a secret and a user account with a known username and password.
3. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy.
2. If no SSID objects exist, perform an AddObject RPC on the Device.Wifi.SSID. partial path.
    a. Record the instance number received in the AddObjectResponse.
3. If no AccessPoint objects exist, perform an AddObject RPC on the Device.Wifi.AccessPoint. partial path
    a. Record the instance number received in the AddObjectResponse
4. Configure the Wifi settings on the CPE using a single SetParameterValues.
5. Reboot the CPE.
6. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.
7. The End Device connects to the CPE's newly created wifi access point, authenticated from the RADIUS Server.
8. Stimulate network traffic across the Wifi connection
9. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.

***Test Metrics:***

1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. Verify CPE sends a Radius request on End Device connection to the Radius server and receives a valid response.
4. The End Device can connect to the wifi connection.
5. The End Device has appropriate network access through the wifi connection.
6. The Stats object has updated to reflect the network traffic across the wifi connection.

### 5.6.10 Wi-Fi Setup WPA2 Personal (Device:2 only)

***Purpose:***

To verify that an ACS can set a valid WiFi configuration on the CPE using WPA2-Personal encryption.

***References:***

Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

ModesSupported includes “WPA2-Personal”

***Parameters:***

The following parameters are required to be implemented for this test.

| **Device.WiFi.Radio.{i}.** | |
|---|---|
| **Enable** | true |
| **Channel** | <unsignedInt> |
| **OperatingFrequencyBand** | 2.4GHz or 5GHz |
| **OperatingStandards** | Auto |
| **Device.WiFi.SSID.{i}.** | |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDAdvertisementEnabled** | true |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.AccessPoint.{i}.Security.** | |
| **ModeEnabled** | WPA2-Personal |
| **ModesSupported** | Returned from device |
| **PreSharedKey** | hexadecimal string |
| **Device.WiFi.Radio.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy.

2. If no SSID objects exist, perform an AddObject RPC on the Device.Wifi.SSID. partial path.
    a. Record the instance number received in the AddObjectResponse.
3. If no AccessPoint objects exist, perform an AddObject RPC on the Device.Wifi.AccessPoint. partial path
    a. Record the instance number received in the AddObjectResponse
4. Configure the Wifi settings on the CPE using a single SetParameterValues.
5. Reboot the CPE.
6. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.
7. The End Device connects to the CPE's newly created wifi access point.
8. Stimulate network traffic across the Wifi connection
9. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.

***Test Metrics:***
1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. The End Device can connect to the wifi connection.
4. The End Device has appropriate network access through the wifi connection.
5. The Stats object has updated to reflect the network traffic across the wifi connection.

### 5.6.11 Wi-Fi Setup WPA2 Enterprise (Device:2 Only)

***Purpose:***

To verify that an ACS can set a valid Wi-Fi configuration on the CPE using WPA2-Enterprise Encryption.

***References:***

Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

ModesSupported includes "WPA2-Enterprise"

***Parameters:***

The following parameters are required to be implemented for this test.

| **Device.WiFi.Radio.{i}.** | |
|---|---|
| **Enable** | true |
| **Channel** | <unsignedInt> |
| **OperatingFrequencyBand** | 2.4GHz or 5GHz |
| **OperatingStandards** | Auto |
| **Device.WiFi.SSID.{i}.** | |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDAdvertisementEnabled** | true |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.AccessPoint.{i}.Security.** | |
| **ModeEnabled** | WPA2-Enterprise |
| **RadiusServerIPAddr** | <IP address of RADIUS server> |
| **RadiusServerPort** | <Port of RADIUS Server> |
| **RadiusSecret** | <string> |
| **Device.WiFi.Radio.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a RADIUS Server set up configured with a secret and a user account with a known username and password.
3. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy.
2. If no SSID objects exist, perform an AddObject RPC on the Device.Wifi.SSID. partial path.
   a. Record the instance number received in the AddObjectResponse.
3. If no AccessPoint objects exist, perform an AddObject RPC on the Device.Wifi.AccessPoint. partial path
   a. Record the instance number received in the AddObjectResponse
4. Configure the Wifi settings on the CPE using a single SetParameterValues.
5. Reboot the CPE.
6. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.
7. The End Device connects to the CPE's newly created wifi access point, authenticated from the RADIUS Server.
8. Stimulate network traffic across the Wifi connection
9. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.

***Test Metrics:***

1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. Verify CPE sends a Radius request on End Device connection to the Radius server and receives a valid response.
4. The End Device can connect to the wifi connection.
5. The End Device has appropriate network access through the wifi connection.
6. The Stats object has updated to reflect the network traffic across the wifi connection.

### 5.6.12 Wi-Fi Setup WPA-WPA2 Personal (Device:2 Only)

***Purpose:***

To verify that an ACS can set a valid WiFi configuration on the CPE using WPA-WPA2-Personal encryption.

***References:***

Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

ModesSupported includes "WPA-WPA2-Personal"

***Parameters:***

The following parameters are required to be implemented for this test.

| **Device.WiFi.Radio.{i}.** | |
|---|---|
| **Enable** | true |
| **Channel** | <unsignedInt> |
| **OperatingFrequencyBand** | 2.4GHz or 5GHz |
| **OperatingStandards** | Auto |
| **Device.WiFi.SSID.{i}.** | |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDAdvertisementEnabled** | true |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.AccessPoint.{i}.Security.** | |
| **ModeEnabled** | WPA-WPA2-Personal |
| **ModesSupported** | Returned from device |
| **PreSharedKey** | hexadecimal string |
| **Device.WiFi.Radio.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy.

2. If no SSID objects exist, perform an AddObject RPC on the Device.Wifi.SSID. partial path.
    a. Record the instance number received in the AddObjectResponse.
3. If no AccessPoint objects exist, perform an AddObject RPC on the Device.Wifi.AccessPoint. partial path
    a. Record the instance number received in the AddObjectResponse
4. Configure the Wifi settings on the CPE using a single SetParameterValues.
5. Reboot the CPE.
6. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.
7. The End Device connects to the CPE's newly created wifi access point.
8. Stimulate network traffic across the Wifi connection
9. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.

***Test Metrics:***

1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. The End Device can connect to the wifi connection.
4. The End Device has appropriate network access through the wifi connection.
5. The Stats object has updated to reflect the network traffic across the wifi connection.

### 5.6.13 Wi-Fi Setup WPA-WPA2 Enterprise (Device:2 Only)

***Purpose:***

To verify that an ACS can set a valid Wi-Fi configuration on the CPE using WPA-WPA2-Enterprise Encryption.

***References:***

Device:2.0

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

ModesSupported includes “WPA-WPA2-Enterprise”

***Parameters:***

The following parameters are required to be implemented for this test.

| **Device.WiFi.Radio.{i}.** | |
|---|---|
| **Enable** | true |
| **Channel** | <unsignedInt> |
| **OperatingFrequencyBand** | 2.4GHz or 5GHz |
| **OperatingStandards** | Auto |
| **Device.WiFi.SSID.{i}.** | |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDAdvertisementEnabled** | true |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.AccessPoint.{i}.Security.** | |
| **ModeEnabled** | WPA-WPA2-Enterprise |
| **ModesSupported** | Returned from device |
| **RadiusServerIPAddr** | <IP address of RADIUS server> |
| **RadiusServerPort** | <Port of RADIUS Server> |
| **RadiusSecret** | <string> |
| **Device.WiFi.Radio.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a RADIUS Server set up configured with a secret and a user account with a known username and password.
3. Have an End Device with the ability to connect to wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy.
2. If no SSID objects exist, perform an AddObject RPC on the Device.Wifi.SSID. partial path.
    a. Record the instance number received in the AddObjectResponse.
3. If no AccessPoint objects exist, perform an AddObject RPC on the Device.Wifi.AccessPoint. partial path
    a. Record the instance number received in the AddObjectResponse
4. Configure the Wifi settings on the CPE using a single SetParameterValues.
5. Reboot the CPE.
6. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.
7. The End Device connects to the CPE's newly created wifi access point.
8. Stimulate network traffic across the Wifi connection
9. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.

***Test Metrics:***

1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. The End Device can connect to the wifi connection.
4. The End Device has appropriate network access through the wifi connection.
5. The Stats object has updated to reflect the network traffic across the wifi connection.

### 5.6.14 Wi-Fi Setup – Add SSID (Device:2 Only)

***Purpose:***

To verify that an ACS can add an SSID object successfully to a preexisting Wifi configuration and that the new SSID is able to be successfully used to gain appropriate network access.

***References:***

Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test.

| **Device.WiFi.SSID.{i}.** | |
|---|---|
| **Enable** | true |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.SSID.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a pre-existing, working WiFi Configuration on the CPE.
3. Have an End Device with the ability to connect to existing wireless networks.

***Procedure:***

1. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy. Take special note of the existing SSID object's LowerLayers parameter (Device.Wifi.SSID.{j}.LowerLayers)
2. Perform an AddObject RPC on “Device.Wifi.SSID.” and record the instance number and status returned.
3. If the status of the Add Object RPC returned a 1, reboot the device.
4. Perform a Get Parameter Values RPC on the path from Procedure step 2 and confirm the object was added successfully.

5. Peform an AddObject RPC on “Device.WiFi.AccessPoint.” and record the instance number and status returned.
6. If the status of the Add Object RPC returned a 1, reboot the device.
7. Perform a Get Parameter Values RPC on the path from Procedure step 5 and confirm the object was added successfully.
8. Configure the Wifi settings on the CPE to utilize the new SSID object using the SetParameterValues RPC:
    a. Device.Wifi.SSID.{i}.LowerLayers = <Value returned in Procedure step 1>
    b. Device.Wifi.SSID.{i}.Enable = true
    c. Device.Wifi.SSID.{i}.SSID = <Value different from Device.Wifi.SSID.{j}.SSID>
    d. Device.Wifi.AccessPoint.{k}.SSIDReference = Device.Wifi.SSID.{i}
9. The End Device connects to the CPE’s newly created WiFi SSID using the same authentication method and credentials that the preexisting configuration uses.
10. Reboot the CPE
11. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.
12. The End Device connects to the CPE’s newly created wifi SSID using the same authentication method and credentials that the preexisting configuration uses.
13. Stimulate network traffic across the Wifi connection.
14. Perform a GetParameterValues on the Stats object listed in the Parameters section and note the values.

***Test Metrics:***

1. Verify that the SetParameterValuesResponse is valid.
2. Settings persist across reboot.
3. The End Device can connect to the new SSID.
4. The End Device has appropriate network access using the new SSID.
5. The Stats object has updated to reflect the network traffic across the WiFi connection.

### 5.6.15 Wi-Fi Setup – Remove SSID (Device:2 Only)

***Purpose:***

To verify that an ACS can remove an SSID object successfully from a preexisting Wifi configuration and that the new SSID is able to be successfully used to gain appropriate network access.

***References:***

Device:2.0

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| WiFiRadio:1, WiFiSSID:1, WiFiAccessPoint:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test.

| **Device.WiFi.SSID.{i}.** | |
|---|---|
| **Enable** | true |
| **SSID** | <string> |
| **LowerLayers** | Device.WiFi.Radio.{i}. |
| **Device.WiFi.AccessPoint.{i}.** | |
| **SSIDReference** | Device.WiFi.SSID.{i}. |
| **Device.WiFi.SSID.{i}.Stats.** | |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Have a preexisting, working Wifi configuration on the CPE.
3. Have an End Device (Laptop) with the ability to connect to wireless networks.

***Procedure:***

1. Verify that the existing Wifi configuration is operating correctly by connecting the End Device to it. Note the SSID that was connected to.
2. Perform a GetParameterValues on the root partial path listed in the parameters section to learn the existing stack hierarchy. Note the instance number of the SSID object with the SSID field matching the one connected to in Procedure Step 1 as well as the instance number of the AccessPoint with the SSIDReference field mapped to the appropriate SSID.

3. Peform a DeleteObject RPC on the Device.Wifi.SSID.{instance noted in Procedure Step 2} and Device.Wifi.AccessPoint.{instance noted in Procedure Step 2}, noting the returned value.
4. If the status returned is 1, reboot the CPE.
5. Perform a GetParameterValues on Device.Wifi.SSID. And Device.Wifi.AccessPoint. to confirm deletion.
6. Attempt to re-connect the End Device to the removed SSID.
7. If status returned in the DeleteObject was 0, reboot the CPE and then perform a GetParameterValues on Device.Wifi.SSID. And Device.Wifi.AccessPoint. to confirm deletion persisted across reboot.

***Test Metrics:***

1. End Device can initially connect to the SSID
2. Deletion of SSID object persists across reboot.
3. The End Device cannot connect to the deleted SSID.

## 5.7 Localized Strings

### 5.7.1 Non-Printable ASCII Characters in a SetParameterValues RPC (Device:2 Only)

***Purpose:***

To verify that an ACS can configure localized strings in the device.

***References:***

Device:2.0

Section 4.4\REC-xml

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Baseline:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| Device.DeviceInfo. | |
|---|---|
| **ProvisioningCode** | <string> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

***Procedure:***

1. ACS triggers a new session with the CPE
2. The CPE establishes the session
3. ACS performs a SPV on the above parameter and sets a new value, which is randomly generated and fulfills these criteria:
    a. The string contains non-printable ASCII characters
    b. The string contains XML characters, which have to be escaped like “<,>, &”
4. ACS ends the session
5. ACS triggers a new session with the CPE
6. The CPE establishes the session
7. ACS performs a GPV on the above parameter
8. ACS ends the session
9. Repeat steps 1-8 with a new randomly generated string.
10. Repeat steps 1-8 with a new randomly generated string.

***Test Metrics:***

1. Validate that both session can be successfully executed
2. Validate that the GPV and SPV are successful

3. Validate that the string returned in step 7 matches the string set in step 3
4. Validate that the string is properly encoded in step 3 and step 7

### 5.7.2 Multi-Byte Encoding in a SetParameterValues RPC (Device:2 Only)

***Purpose:***

To verify that an ACS can configure strings in the CPE that are 64 characters but more than 64 bytes.

***References:***

Device:2.0

Section 4.3.3\REC-xml

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Baseline:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| Device.DeviceInfo. | |
|---|---|
| **ProvisioningCode** | <string> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

***Procedure:***

1. ACS triggers a new session with the CPE
2. The CPE establishes the session
3. ACS performs a SPV on the above parameter and sets a new value, which is randomly generated and fulfills these criteria:
    a. The string uses a multi-byte encoding
    b. The string contains 64 characters
4. ACS ends the session
5. ACS triggers a new session with the CPE
6. The CPE establishes the session
7. ACS performs a GPV on the above parameter
8. ACS ends the session

***Test Metrics:***

1. Validate that both session can be successfully executed
2. Validate that the GPV and SPV are successful
3. Validate that the string returned in step 7 matches the string set in step 3
4. Validate that the string is properly encoded in step 3 and step 7

### 5.7.3 Non-ASCII Characters in a ParameterKey (Device:2 Only)

***Purpose:***

To verify that an ACS can configure localized strings in the device.

***References:***

Device:2.0

Section 4.4\REC-xml

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Baseline:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| Device.DeviceInfo. | |
|---|---|
| **ProvisioningCode** | <string> |

| Device.ManagementServer. | |
|---|---|
| **ParameterKey** | <string> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

***Procedure:***

1. ACS triggers a new session with the CPE.
2. The CPE establishes the session.
3. ACS performs a SetParameterValues RPC on the ProvisioningCode and sets a new value. In the SPV the ACS uses a ParameterKey, which is randomly generated and fulfills these criteria:
    a. The string contains non-printable ASCII characters
    b. The string contains XML characters, which have to be escaped like “<,>, &”.
4. ACS ends the session.
5. ACS triggers a new session with the CPE.
6. The CPE establishes the session.
7. ACS performs a GetParameterValues on ParameterKey.
8. ACS ends the session.
9. Repeat steps 1-8 with a new randomly generated string.
10. Repeat steps 1-8 with a new randomly generated string.

***Test Metrics:***

1. Validate that both session can be successfully executed.
2. Validate that the GetParameterValues RPC and the SetParameterValues RPC are successful.
3. Validate that the string returned in step 7 matches the string set in step 3.
4. Validate that the string is properly encoded in step 3 and step 7.

### 5.7.4 Multi-Byte Encoding in ParameterKey (Device:2 Only)

***Purpose:***

To verify that an ACS can configure strings in the CPE that are 32 characters but more than 32 bytes.

***References:***

Device:2.0

Section 4.3.3\REC-xml

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Baseline:1 | N/A |

***Optional Features***

None

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.DeviceInfo.** | |
|---|---|
| **ProvisioningCode** | <string> |

| **Device.ManagementServer.** | |
|---|---|
| **ParameterKey** | <string> |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

***Procedure:***

1. ACS triggers a new session with the CPE
2. The CPE establishes the session
3. ACS performs a SetParameterValues RPC on the ProvisioningCode and sets a new value. In the SPV the ACS uses a ParameterKey, which is randomly generated and fulfills these criteria:
    a. The string uses a multi-byte encoding
    b. The string contains 32 characters
4. ACS ends the session
5. ACS triggers a new session with the CPE
6. The CPE establishes the session
7. ACS performs a GetParameterValues RPC on ParameterKey.
8. ACS ends the session

***Test Metrics:***

1. Validate that both session can be successfully executed

2. Validate that the GetParameterValues RPC and the SetParameterValues RPC are successful
3. Validate that the string returned in step 7 matches the string set in step 3
4. Validate that the string is properly encoded in step 3 and step 7

### 5.7.5 Non-ASCII Characters in CommandKey (Device:2 Only)

***Purpose:***
To verify that an ACS can configure localized strings in the device.

***References:***
Device:2.0
Section 4.4\REC-xml

***Profiles***

| **Device:2** | **InternetGatewayDevice:1** |
|---|---|
| Baseline:1 | N/A |

***Optional Features***
None

***Parameters:***
None.

***Test Setup:***
1. Refer to Common Test Setup for setup steps.

***Procedure:***
1. ACS triggers a new session with the CPE
2. The CPE establishes the session
3. ACS performs a Reboot RPC with a Commandkey parameter which is randomly generated and fulfills these criteria:
    a. The string contains non-printable ASCII characters.
    b. The string contains XML characters, which have to be escaped like "<,>, &"
4. ACS ends the session
5. The CPE performs a reboot and reports the CommandKey to the ACS.
6. Repeat steps 1-5 with a new randomly generated string.
7. Repeat steps 1-5 with a new randomly generated string.

***Test Metrics:***
1. Validate that both session can be successfully executed
2. Validate that the Reboot RPC is successful.
3. Validate that the string returned in step 5 matches the string set in step 3 after escaping.
4. Validate that the string is properly encoded in step 3 and step 5

### 5.7.6 Multi-Byte Encoding in CommandKey (Device:2 Only)

***Purpose:***

To verify that an ACS can configure strings in the CPE that are 32 characters but more than 32 bytes.

***References:***

Device:2.0

Section 4.3.3\REC-xml

***Profiles***

| Device:2 | InternetGatewayDevice:1 |
|---|---|
| Baseline:1 | N/A |

***Optional Features***

None

***Parameters:***

None.

***Test Setup:***

1. Refer to Common Test Setup for setup steps.

***Procedure:***

1. ACS triggers a new session with the CPE
2. The CPE establishes the session
3. ACS performs a Reboot RPC with a Commandkey parameter which is randomly generated and fulfills these criteria:
    a. The string uses a multi-byte encoding using UTF-8.
    b. The string contains 32 characters
4. ACS ends the session
5. The CPE performs a reboot and reports the CommandKey to the ACS.

***Test Metrics:***

1. Validate that both session can be successfully executed
2. Validate that the Reboot RPC is successful
3. Validate that the string returned in step 5 matches the string set in step 3 after escaping.
4. Validate that the string is properly encoded in step 3 and step 5

## Future Test Cases

### I.1 Configuration Backup and Restore – Incorrect Backup File

***Purpose:***
This test is designed to verify that the CPE responds to invalid restoration of a configuration with a correct fault code.

***References:***
Section A.4.1.5/TR-069a1 or later

***Parameters:***
These parameters are for the Download RPC:

Download parameters:
CommandKey: a unique value
FileType: "3 Vendor Configuration File"
URL: location of the invalid file
Username: used to authenticate CPE with file server
Password: used to authenticate CPE with file server
FileSize: size of the file in bytes
TargetFileName: name of file
DelaySeconds: 0
SuccessURL: Empty string
FailureURL: Empty string

***Test Setup:***
1. ACS connected to the network
2. CPE connected to the network and configured with an ACS URL that corresponds to the ACS in #1
3. Have device to capture traffic between the device and both the file server and ACS
4. Establish a server to download files.

***Procedure:***
1. ACS schedules a download RPC on the CPE with the parameters above
2. Allow CPE to respond DownloadResponse or TransferComplete

***Test Metrics:***
3. Verify that the CPE responds to the Download RPC with a DownloadResponse or TransferComplete with one of the following fault codes: 9003 (Invalid argument),

## I.2 DHCP Provisioning – Disable DHCP

***Purpose:***

To verify that an ACS can configure the basic DHCPv4 configuration required on a CPE device.

***References:***

TR-181i2a7

***Parameters:***

The following parameters within the InternetGatewayDevice.LANDevice.{i}.LANHostConfigManagement. table (for devices that implement the InternetGatewayDevice:1 root data model) and Device.DHCPv4.Server.Pool.{i}. table (for devices that implement the Device:2 root data model) are required to be implemented for this test:

| **Name** | **InternetGatewayDevice:1** | **Device:2** | **Value** |
|---|---|---|---|
| Server Configurable | DHCPServerConfigurable | -- | TRUE |
| Server Enable | DHCPServerEnable | Disable | FALSE |
| Interface | -- | Interface | Returned from Device |
| Minimum Address | MinAddress | MinAddress | <minimum address> |
| Maximum Address | MaxAddress | MaxAddress | <maximum address> |
| Lease Time | DHCPLeaseTime | -- | 60 |

***Test Setup:***

1. ACS connected to the network
2. CPE connected to the network and configured with an ACS URL that corresponds to the ACS in #1
3. CPE supports the appropriate DHCP parameters in the parameters section
4. If a Device CPE, verify that the Interface parameter is a valid path name in the IP Interface table.
5. Have an end system that can attempt to connect to the CPE via DHCP.

***Procedure:***

1. If the DUT implements the Device:2 root data model, perform an AddObject RPC on Device.Server.Pool., recording the InstanceNumber value returned in the AddObjectResponse

2. On the ACS schedule a SetParameterValues RPC on the appropriate Configurable parameters in the Parameters Section
3. Allow the end system to connect to the CPE via DHCP.
4. Perform a reboot of the CPE
5. Allow the end system to attempt connect to the CPE via DHCP

***Test Metrics:***

1. Verify that the SetParameterValuesResponse RPC is correct for all parameters
2. Verify that the end system doesn't connect to the CPE within the designated range.
3. Verify that that end system doesn't connects to the CPE within the designated range after the reboot

## I.3 Diagnostics IPPing – Error Condition

***Purpose:***

To verify that an ACS can perform an IP Ping Diagnostics test on the CPE where the CPE returns a Error_NoRouteToHost.

***References:***

InternetGatewayDevice:1.6 and Device:2.6

***Parameters:***

The following parameters within the InternetGatewayDevice.IPPingDiagnostics. table (for devices that implement the InternetGatewayDevice:1 data model) or Device.IP.Diagnostics.IPPing. table (for devices that implement the Device:2 data model) are required to be implemented for this test:

| **Name** | **InternetGatewayDevice:1** | **Device:2** | **Value** |
|---|---|---|---|
| Diagnostics State | DiagnosticsState | DiagnosticsState | Requested |
| Interface | Interface | Interface | Return from device |
| Host | Host | Host | Unknown Host |
| Number of Repetitions | NumberOfRepetitions | NumberOfRepetitions | 3 |
| Time out | Timeout | Timeout | 1000 |
| Data block size | DataBlockSize | | 128 |
| DSCP | DSCP | DSCP | 0 |
| TimeZone | LocalTimeZone | LocalTimeZone | |

The following WAN Interface parameters are to determine the WAN interface of the CPE, InternetGatewayDevice.Layer3Forwarding.Default and Device.IP.Interface:

| **Name** | **InternetGatewayDevice:1** | **Device:2** | **Value** |
|---|---|---|---|
| Connection | DefaultConnectionService | Interface | Returned from Device |

The following Diagnostics parameters are to determine the state of the IP ping test of the CPE, InternetGatewayDevice.IPPingDiagnostics and Device.IP.Diagnostics.IPPing:

| **Name** | **InternetGatewayDevice:1** | **Device:2** | **Value** |
|---|---|---|---|
| DiagnosticsState | DiagnosticsState | DiagnosticsState | Returned from Device |

***Test Setup:***

1. ACS connected to the network
2. CPE connected to the network and configured with an ACS URL that corresponds to the ACS in Setup #1
3. Wireshark running between ACS and CPE.

***Procedure:***

1. ACS performs a GetParameterValues RPC on the WAN Interface parameter to determine the current interface in the parameter section above.
2. ACS performs a SetParameterValues RPC on the ping diagnostics parameters in the parameter section above.
3. ACS performs a GetParameterValues on the InternetGatewayDevice.IPPingDiagnostics or Device.IP.Diagnostics.IPPing to determine results of ping test

***Test Metrics:***

1. Validate that an Inform RPC is sent from the ACS with Event Code “8 DIAGNOSTICS COMPLETE”
2. Validate that the DiagnosticsState is set to Error_NoRouteToHost.

## I.4 TraceRoute Diagnostics – Error Condition

***Purpose:***

This test is designed to validate that the CPE supports TraceRoute diagnostics test and reports results appropriately to the ACS.

***References:***

InternetGatewayDevice:1.6 and Device:2.6

***Parameters:***

The following parameters with the InternetGatewayDevice.TraceRouteDiagnostics. table (for devices that implement the InternetGatewayDevice:1 data model) and Device.IP.Diagnostics.TraceRoute. (for devices that implement the Device:2 data model) are required to be implemented for this test:

| **Name** | **InternetGatewayDevice:1** | **Device:2** | **Value** |
|---|---|---|---|
| Interface | Interface | | Return from device |
| Host | Host | Host | Ping host |
| Number of Tries | NumberOfTries | | 3 |
| Time out | Timeout | Timeout | 5000 |
| Data block size | DataBlockSize | DataBlockSize | 128 |
| DSCP | DSCP | DSCP | 0 |
| Max Hop Count | MaxHopCount | MaxHopCount | 1 |
| Diagnostic State | DiagnosticsState | DiagnosticsState | |

***Test Setup:***

1. ACS connected to the network
2. CPE connected to the network and configured with an ACS URL that corresponds to the ACS in Setup #1
3. Wireshark running between ACS and CPE.
4. Ensure there are at least 2 hops between the CPE and Host

***Procedure:***

1. Establish a CWMP session between the CWMP Analyzer and DUT with successful Inform exchanges.
2. Schedule a GetParameterValues RPC on the current WAN interface.
3. Schedule a SetParameterValues RPC on the DUT on the trace route diagnostic parameters listed above.
4. Schedule a GetParameterValues RPC on the DUT on Diagnostic State.

***Test Metrics:***

1. The DUT can properly respond to the GetParameterValues request on WAN interface.
2. The DUT is able to properly respond to the SetParameterValues for diagnostics parameter.
3. Validate that the DiagnosticsState is set to “Error_MaxHopCountExceeded”

## I.5 IPPing Diagnostics – Periodic Inform

***Purpose:***

To verify that an ACS can perform an IP Ping Diagnostics test on the CPE and receive a periodic inform without an Diagnostic complete event before the diagnostic completes.

***References:***

InternetGatewayDevice:1.6 and Device:2.6

***Parameters:***

The following parameters within the InternetGatewayDevice.IPPingDiagnostics. table (for devices that implement the InternetGatewayDevice:1 data model) or Device.IP.Diagnostics.IPPing. table (for devices that implement the Device:2 data model) are required to be implemented for this test:

| **Name** | **InternetGatewayDevice:1** | **Device:2** | **Value** |
|---|---|---|---|
| Diagnostics State | DiagnosticsState | DiagnosticsState | Requested |
| Interface | Interface | Interface | Return from device |
| Host | Host | Host | Unknown Host |
| Number of Repetitions | NumberOfRepetitions | NumberOfRepetitions | 10000 |
| Time out | Timeout | Timeout | 100000 |
| Data block size | DataBlockSize | | 128 |
| DSCP | DSCP | DSCP | 0 |
| TimeZone | LocalTimeZone | LocalTimeZone | |

The following WAN Interface parameters are to determine the WAN interface of the CPE, InternetGatewayDevice.Layer3Forwarding.Default and Device.IP.Interface:

| **Name** | **InternetGatewayDevice:1** | **Device:2** | **Value** |
|---|---|---|---|
| Connection | DefaultConnectionService | Interface | Returned from Device |

The following Diagnostics parameters are to determine the state of the IP ping test of the CPE, InternetGatewayDevice.IPPingDiagnostics and Device.IP.Diagnostics.IPPing:

| **Name** | **InternetGatewayDevice:1** | **Device:2** | **Value** |
|---|---|---|---|
| DiagnosticsState | DiagnosticsState | DiagnosticsState | Returned from Device |

The following parameters are to set to ensure the CPE performs a periodic inform prior to the completion of Diagnostic, InternetGatewayDevice.ManagementServer and Device.IP.Diagnostics.ManagementServer:

| **Name** | **InternetGatewayDevice:1** | **Device:2** | **Value** |
|---|---|---|---|
| PeriodicInformInterval | PeriodicInformInterval | PeriodicInformInterval | 30 |

***Test Setup:***

1. ACS connected to the network
2. CPE connected to the network and configured with an ACS URL that corresponds to the ACS in Setup #1
3. Wireshark running between ACS and CPE.

***Procedure:***

1. ACS performs a SetParameterValues RPC on the ManagementServer parameters in the parameter section above.
2. ACS performs a GetParameterValues RPC on the WAN Interface parameter to determine the current interface in the parameter section above.
3. ACS performs a SetParameterValues RPC on the ping diagnostics parameters in the parameter section above.
4. ACS performs a GetParameterValues on the InternetGatewayDevice.IPPingDiagnostics or Device.IP.Diagnostics.IPPing to determine results of ping test

***Test Metrics:***

1. During diagnostic: Validate that the Inform RPC does not have a “8 DIAGNOSTICS COMPLETE” Event Code
2. After diagnostic:-Validate that an Inform RPC is sent from the ACS with Event Code “8 DIAGNOSTICS COMPLETE”

## I.6 Deny/Allow Inbound IPv6 (Device:2 Only)

***Purpose:***

To verify the ACS can configure a firewall on the CPE that allows certain IPv6 traffic from the WAN side.

***References:***

Device:2.2

***Parameters:***

The following parameters are required to be implemented for this test:

| **Device.Firewall.** | |
|---|---|
| **Enable** | true |
| **Config** | Advanced |
| **AdvancedLevel** | Device.Firewall.Level.{i}. |
| **Device.Firewall.Level.{i}.** | |
| **Chain** | Returned from device |
| **DefaultPolicy** | Drop |
| **Device.Firewall.Chain.{i}.** | |
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **DestIP** | IPv6 Address |
| **SourceIP** | IPv6 Address |
| **Protocol** | int[-1:255] |
| **DestPort** | int[-1:65535] |

***Test Setup:***

1. Refer to Common Test Setup for setup steps.
2. Refer to Determine WAN Interface to determine the WAN interface.
3. Two Devices that can send and receive TCP and UDP connections over IPv6 connected to the LAN side of the CPE (End Device 1 and End Device 2)
4. Device that can send and receive TCP and UDP connections over IPv6 connected on the WAN side of the CPE (WAN Device)
5. Have a Network Analyzer to capture traffic between the CPE and the End Device.

***Procedure:***

1. Configure ACS to send an AddObject RPC for Device.Firewall.Level and record the returned instance number.
2. Configure ACS to send a GetParameterValues RPC for Device.Firewall.Level.{i}.Chain.
3. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule using the path returned in step 2 and record the returned instance number.

4. ACS performs a SetParameterValues RPC on the following parameters using the instance numbers returned in steps 1-3.

| **Device.Firewall.** | |
|---|---|
| **Config** | Advanced |
| **AdvancedLevel** | Device.Firewall.Level.{i}. |
| **Device.Firewall.Level.{i}.** | |
| **Chain** | Returned from device |
| **DefaultPolicy** | Drop |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestIP** | <Unused IP Address> |
| **SourceIP** | <Unused IP Address> |

5. ACS enables the Firewall by sending a SetParameterValues RPC for Device.Firewall.Enable with the value set to “true.”
6. WAN Device attempts to open a TCP connection over IPv6, destination port 25, with End Device 1.
7. WAN Device attempts to open a TCP connection over IPv6, destination port 25, with End Device 2.
8. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule and record the returned instance number.
9. ACS Configures the new Firewall Rule by sending a SetParameterValues RPC with the following name/value pairs using the instance number returned in step 8.

| **Device.Firewall.Chain.{i}.** | |
|---|---|
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **Protocol** | 6 |
| **DestPort** | 25 |
| **DestIP** | End Device1 IP |

10. WAN Device attempts to open a TCP connection over IPv6, destination port 25, with End Device 1.
11. WAN Device attempts to open a TCP connection over IPv6, destination port 25, with End Device 2.
12. Configure ACS to send an AddObject RPC for Device.Firewall.Chain.{i}.Rule and record the returned instance number.
13. ACS Configures the new Firewall Rule by sending a SetParameterValues RPC with the following name/value pairs using the instance number returned in step 12.

| **Device.Firewall.Chain.{i}.** | |
|---|---|
| **Enable** | true |
| **Device.Firewall.Chain.{i}.Rule.{i}.** | |

| | |
|---|---|
| **Target** | Accept |
| **DestInterface** | Default WAN Interface |
| **Protocol** | 6 |
| **DestPort** | 25 |
| **DestIP** | End Device2 IP |

14. WAN Device attempts to open a TCP connection over IPv6, destination port 25, with End Device 1.
15. WAN Device attempts to open a TCP connection over IPv6, destination port 25, with End Device 2.

***Test Metrics:***

1. In Step 6, CPE does not forward TCP traffic from WAN Device to End Device1.
2. In Step 7, CPE does not forward TCP traffic from WAN Device to End Device2.
3. In Step 10, CPE forwards TCP traffic from WAN Device to End Device1.
4. In Step 11, CPE does not forward TCP traffic from WAN Device to End Device2.
5. In Step 14, CPE forwards TCP traffic from WAN Device to End Device1.
6. In Step 15, CPE forwards TCP traffic from WAN Device to End Device2.

## I.7 Download Diagnostics over HTTPS – TotalBytesReceived

***Purpose:***

To verify that an ACS and CPE can interoperate while performing the download diagnostics function over HTTPS. This test will be run if supported on the CPE.

***References:***

InternetGatewayDevice.1.3
Device:2.0

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.IP.Diagnostics.DownloadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **DownloadURL** | <HTTPS URL of file on server> |
| **TestBytesReceived** | Returned from device |
| **TotalBytesReceived** | Returned from device |
| **DownloadTransports** | Returned from device |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.DownloadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **DownloadURL** | <HTTPS URL of file on server> |
| **TestBytesReceived** | Returned from device |
| **TotalBytesReceived** | Returned from device |
| **InternetGatewayDevice.Capabilities.PerformanceDiagnostics.** | |
| **DownloadTransports** | Returned from device |

The DownloadTransports parameter must include HTTP for this test to be executed.

***Test Setup:***

7. Refer to Common Test Setup for setup steps.
8. Have an HTTP server and file to perform the download over HTTPS. The file should be big enough for the file transfer to last at least 30 seconds.

***Procedure:***

9. On the ACS schedule a SetParameterValues RPC on the Interface, DownloadURL and DiagnosticsState parameters listed in the parameters section.
10. Allow the CPE to perform the specific diagnostic tests and send an Inform message with an event code of “8 DIAGNOSTICS COMPLETE”.
11. On the ACS schedule a GetParameterValues RPC on the DownloadDiagnostics object.

12. Perform a reboot of the CPE.
13. On the ACS schedule a GetParameterValues RPC on the DownloadDiagnostics object.

***Test Metrics:***

14. Verify that the SetParameterValuesResponse RPC is correct
15. Verify that an Inform message is received with an event code of "8 DIAGNOSTICS COMPLETE"
16. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
17. Verify that TestBytesReceived is correct.
18. Verify that EOMTime – BOMTime is within 1 second of the time that the file server reports the transfer took.
19. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
    a. If the value is "Completed" verify that the values in the DownloadDiagnostic object are the same as they were before the reboot.
20. Verify that TotalBytesReceived is within 1% of the measured traffic.

## I.8 Upload Diagnostics over HTTP - TotalBytesSent

***Purpose:***

To verify that an ACS and CPE can inter-operate while performing the upload diagnostics function over HTTP. This test will be run if supported on the CPE.

***References:***

InternetGatewayDevice.1.3
Device:2.0

***Parameters:***

The following parameters are required to be implemented for this test:

For CPE that implement the Device:2 root data model:

| **Device.IP.Diagnostics.UploadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **UploadURL** | <HTTPS URL of server> |
| **TestFileLength** | <value long enough for the upload to last 30 seconds> |
| **TotalBytesSent** | Returned from device |
| **UploadTransports** | Returned from device |

For CPE that implement the InternetGatewayDevice:1 root data model:

| **InternetGatewayDevice.UploadDiagnostics.** | |
|---|---|
| **DiagnosticsState** | Requested |
| **Interface** | Returned from device |
| **UploadURL** | <HTTPS URL of server> |
| **TestFileLength** | <value long enough for the upload to last 30 seconds> |
| **TotalBytesSent** | Returned from device |
| **InternetGatewayDevice.Capabilities.PerformanceDiagnostics.** | |
| **UploadTransports** | Returned from device |

The UploadTransports parameter must include HTTP for this test to be executed.

***Test Setup:***

21. Refer to Common Test Setup for setup steps.
22. Have an HTTP server to perform the Upload over HTTPS.
23. Wireshark running between ACS and CPE.

***Procedure:***

24. On the ACS schedule a SetParameterValues RPC on the Interface, UploadURL and DiagnosticsState parameters listed in the parameters section.
25. Allow the CPE to perform the specific diagnostic tests and send an Inform message with an event code of “8 DIAGNOSTICS COMPLETE”.
26. On the ACS schedule a GetParameterValues RPC on the UploadDiagnostics object.

27. Perform a reboot of the CPE
28. On the ACS schedule a GetParameterValues RPC on the UploadDiagnostics object.

***Test Metrics:***

29. Verify that the SetParameterValuesResponse RPC is correct
30. Verify that an Inform message is received with an event code of “8 DIAGNOSTICS COMPLETE”
31. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
32. Verify that the test server received a file of TestLengthBytes size.
33. Verify that EOMTime – BOMTime is within 1 second of the time that the file server reports the transfer took.
34. Verify that the GetParameterValuesResponse contains a valid value for the DiagnosticsState parameter.
    a. If the value is “Completed” verify that the values in the UploadDiagnostic object are the same as they were before the reboot.
35. Verify that TotalBytesSent is within 1% of the measured traffic.

## Open Issues

| Item # | Item Description | Reference(s) | Status |
|---|---|---|---|
| Item x | | | |

End of Broadband Forum Test Plan TP-181