# Network Protocols SNP, PXE and BISTest

## EFI\_SIMPLE\_NETWORK\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_SIMPLE\_NETWORK\_PROTOCOL Section..

### Start()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.1.1 | 0x200d5d39, 0x8131, 0x434f, 0x95, 0x89, 0xc6, 0xbe, 0x88, 0x69, 0x5d, 0xf4 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Start - returns EFI\_ALREADY\_STARTED when calling Start() while the network interface is already started | Call Start() when the network interface is already started. The return status should be EFI\_ALREADY\_STARTED and the state should be “Started”. |
| 5.11.1.1.2 | 0xf58651fe, 0x0538, 0x4407, 0x88, 0xe0, 0x88, 0xb8, 0xda, 0x18, 0x38, 0x3a | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Start - returns EFI\_SUCCESS when calling Start()to verify the interface state. | Call Start()The return status should be EFI\_SUCCESS and the interface state should be EfiSimpleNetworkStarted. |

### Stop()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.2.1 | 0xda5a5aea, 0x0a26, 0x4b65, 0x90, 0x84, 0x92, 0x15, 0xc5, 0x43, 0x21, 0xa0 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Stop - Invokes Stop() when the network interface is not started returns EFI\_NOT\_STARTED. | Call Stop() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.2.2 | 0xd0ecac27, 0xfa2e, 0x4b7d, 0x89, 0x2c, 0xc0, 0xff, 0x70, 0x54, 0x13, 0x44 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Stop - Invokes Stop() verifies the interface state and returns EFI\_SUCCESS. | Call Stop(). The return status should be EFI\_SUCCESS and the interface state should be EfiSimpleNetworkStopped. |

### Initialize()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.3.1 | 0xbaa11393, 0x2bfc, 0x43ef, 0xbd, 0xb7, 0x0a, 0xc5, 0x0e, 0x8a, 0x3a, 0x21 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Initialize - Invokes Initialize() when the network interface is not started returns EFI\_NOT\_STARTED. | Call Initialize() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.3.2 | 0x9d4eec8d, 0xdf2f, 0x4f5e, 0x9f, 0x95, 0x7e, 0x51, 0x62, 0xc2, 0x51, 0x0d | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Initialize - Invokes Initialize()to verify the interface state and returns EFI\_SUCCESS. | Call Initialize(). The return status should be EFI\_SUCCESS and the interface state should be EfiSimpleNetworkInitialized. |
| 5.11.1.3.3 | 0x7b547661, 0xa0aa, 0x4041, 0x99, 0xf6, 0xe2, 0x07, 0x31, 0xf7, 0x98, 0x3c | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Initialize - Invokes Initialize() with extra Tx/Rx specified to verify the interface state and returns EFI\_SUCCESS. | Call Initialize() with extra Tx/Rx specified. The return status should be EFI\_SUCCESS and the interface state should be EfiSimpleNetworkInitialized. |

### Reset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.4.1 | 0xf2fed213, 0xb6ad, 0x4edc, 0x96, 0xd7, 0x4a, 0xdc, 0x2e, 0xbd, 0xbb, 0x1e | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Reset - Invokes Reset() when the network interface is not started returns EFI\_NOT\_STARTED. | Call Reset() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.4.2 | 0x30314e89, 0xdb26, 0x4b01, 0x90, 0xf3, 0x04, 0xd3, 0x1b, 0x19, 0xa6, 0x01 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Reset - Invokes Reset() with an ExtendedVerification value of FALSE verifies interface correctness and returns EFI\_SUCCESS. | Call Reset() with an ExtendedVerification value of FALSE. The return status should be EFI\_SUCCESS and the interface mode should be correct. |
| 5.11.1.4.3 | 0xa3135b96, 0xf9c6, 0x45b6, 0xae, 0x87, 0x15, 0xca, 0xae, 0x31, 0x7e, 0xfb | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Reset - Invokes Reset() with an ExtendedVerification value of TRUE verifies interface correctness and returns EFI\_SUCCESS. | Call Reset() with an ExtendedVerification value of TRUE. The return status should be EFI\_SUCCESS and the interface mode should be correct. |

### Shutdown()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.5.1 | 0x09bb5019, 0x1787, 0x4403, 0xb1, 0x2e, 0x91, 0x93, 0x5c, 0xbd, 0x08, 0xe3 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Shutdown - Invokes Shutdown() when the network interface is not started returns EFI\_NOT\_STARTED. | Call Shutdown() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.5.2 | 0x49365eeb, 0xd66c, 0x4109, 0xb0, 0xcf, 0x36, 0xc8, 0x96, 0xc0, 0x07, 0xec | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Shutdown - Invokes Shutdown()verifies the interface state and returns EFI\_SUCCESS. | Call Shutdown(). The return status should be EFI\_SUCCESS and the interface state should be EfiSimpleNetworkStarted. |

### ReceiveFilters()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.6.1 | 0x3f8d8e2a, 0xdbb1, 0x41b8, 0xb9, 0xd9, 0x5f, 0x79, 0x44, 0xf1, 0xd1, 0xf4 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() when the network interface is not started returns EFI\_NOT\_STARTED. | Call ReceiveFilters() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.6.2 | 0x8b4ed1bb, 0xa4a4, 0x45e8, 0xbf, 0x32, 0x0d, 0x0d, 0x6d, 0x0b, 0xd0, 0x2e | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() when the network interface is not initialized returns EFI\_DEVICE\_ERROR. | Call ReceiveFilters() when the network interface is not initialized. The return status should be EFI\_DEVICE\_ERROR. |
| 5.11.1.6.3 | 0xb6f84e0b, 0x286b, 0x44a6, 0xa0, 0xf8, 0x6d, 0x11, 0x89, 0x7d, 0x56, 0x55 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() with an invalid Enable returns EFI\_INVALID\_PARAMETER. | Call ReceiveFilters() with an invalid Enable. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.6.4 | 0xead4b950, 0xf0d6, 0x4195, 0x94, 0xaa, 0x81, 0x92, 0x56, 0x44, 0xb3, 0x2c | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() with an invalid McastFilterCnt returns EFI\_INVALID\_PARAMETER. | Call ReceiveFilters() with an invalid MCastFilterCnt. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.6.5 | 0x4497e853, 0xc54d, 0x409b, 0x85, 0x01, 0xd5, 0xfb, 0xd2, 0x7a, 0x95, 0xdc | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() with MCastFilterCnt not matching MCastFilter returns EFI\_INVALID\_PARAMETER. | Call ReceiveFilters() with MCastFilterCnt not matching MCastFilter. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.6.6 | 0xd82baa78, 0x2bf8, 0x49db, 0xb5, 0x7f, 0x92, 0x2e, 0xe5, 0x79, 0xc3, 0x7a | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() modifies the multicast receive filter mask (Disable Specified bit), verifies interface correctness, and returns EFI\_SUCCESS. | Call ReceiveFilters() to modify the multicast receive filter mask (Disable Specified bit) and verify interface correctness. The return status should be EFI\_SUCCESS. |
| 5.11.1.6.7 | 0x9605c24a, 0x2090, 0x490d, 0x89, 0x4f, 0xfc, 0xb8, 0xc1, 0xb9, 0xd4, 0xf8 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() modifies the multicast receive filter mask (Enable Specified bit), verifies interface correctness, and returns EFI\_SUCCESS. | Call ReceiveFilters() to modify the multicast receive filter mask (Enable Specified bit) and verify interface correctness. The return status should be EFI\_SUCCESS. |
| 5.11.1.6.8 | 0xd9893cd3, 0x7269, 0x4931, 0x9e, 0xe8, 0x81, 0x62, 0x7a, 0x67, 0x45, 0xe9 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() modifies the multicast receive filter masks (Enable and Disable Specified bit together), verifies interface correctness, and returns EFI\_SUCCESS. | Call ReceiveFilters() to modify the multicast receive filter masks (Enable and Disable Specified bit together) and verify interface correctness. The return status should be EFI\_SUCCESS. |
| 5.11.1.6.9 | 0x056e2680, 0xbcc9, 0x460a, 0x94, 0xb4, 0x9a, 0xe2, 0x99, 0xa7, 0x2c, 0x2c | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() modifies the multicast receive filters list, verifies interface correctness, and returns EFI\_SUCCESS. | Call ReceiveFilters() to modify the multicast receive filters list and verify interface correctness. The return status should be EFI\_SUCCESS. |
| 5.11.1.6.10 | 0x2143092e, 0x03dd, 0x4806, 0x9f, 0xd6, 0x08, 0xd4, 0x2b, 0x9a, 0xbf, 0xc6 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.ReceiveFilters - Invokes ReceiveFilters() resets the multicast receive filters list, verifies interface correctness within test case, and returns EFI\_SUCCESS. | Call ReceiveFilters() to reset the multicast receive filters list and verify interface correctness within test case. The return status should be EFI\_SUCCESS. |

### StationAddress()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.7.1 | 0x4235215c, 0xfad0, 0x4865, 0xa9, 0x7b, 0xde, 0xe4, 0xb7, 0xee, 0xef, 0x98 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.StationAddress - Invokes StationAddress() when the network interface is not started returns EFI\_NOT\_STARTED. | Call StationAddress() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.7.2 | 0x9dfe127c, 0x14b0, 0x476d, 0x9d, 0x68, 0x69, 0x08, 0x15, 0x7e, 0x36, 0xa7 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.StationAddress - Invokes StationAddress() when the network interface is not initialized returns EFI\_DEVICE\_ERROR. | Call StationAddress() when the network interface is not initialized. The return status should be EFI\_DEVICE\_ERROR. |
| 5.11.1.7.3 | 0x6c6fb7ad, 0xf89c, 0x45d6, 0xb3, 0xa6, 0x15, 0x34, 0xfd, 0x72, 0xfb, 0x9d | EFI\_SIMPLE\_NETWORK\_PROTOCOL.StationAddress - Invokes StationAddress() with an invalid parameter returns EFI\_INVALID\_PARAMETER. | Call StationAddress() to change the MAC address when the address is not allowed to be changed. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.7.4 | 0x29177bfa, 0x3775, 0x4d5a, 0x97, 0x37, 0x19, 0xd8, 0x34, 0xa7, 0xbb, 0x8e | EFI\_SIMPLE\_NETWORK\_PROTOCOL.StationAddress - Invokes StationAddress() resets MAC Address, verifies interface correctness, and returns EFI\_SUCCESS. | Call StationAddress() to reset MAC Address and verify interface correctness. The return status should be EFI\_SUCCESS. |
| 5.11.1.7.5 | 0xbbbde63c, 0xa6f5, 0x4438, 0x8a, 0x82, 0xb4, 0xdf, 0xe8, 0xe8, 0x48, 0xfd | EFI\_SIMPLE\_NETWORK\_PROTOCOL.StationAddress - Invokes StationAddress() modifies MAC Address, verifies interface correctness, and returns EFI\_SUCCESS. | Call StationAddress() to modify MAC Address and verify interface correctness. The return status should be EFI\_SUCCESS. |

### Statistics()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.8.1 | 0x62a700f1, 0x075f, 0x4cc0, 0x85, 0x12, 0xee, 0x48, 0x0d, 0xbc, 0x69, 0x2c | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Statistics - Invokes Statistics() when the network interface is not started returns EFI\_NOT\_STARTED. | Call Statistics() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.8.2 | 0x71173afd, 0x5dc9, 0x42ea, 0xa8, 0xad, 0x6e, 0xc0, 0x97, 0x7a, 0xdc, 0xa6 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Statistics - Invokes Statistics() when the network interface is not initialized returns EFI\_DEVICE\_ERROR. | Call Statistics() when the network interface is not initialized. The return status should be EFI\_DEVICE\_ERROR. |
| 5.11.1.8.3 | 0x743b75d1, 0xaf66, 0x495c, 0xaf, 0x5a, 0x1d, 0xdf, 0x7f, 0xe4, 0xa6, 0x82 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Statistics - Invokes Statistics() with small buffer returns EFI\_BUFFER\_TOO\_SMALL or EFI\_UNSUPPORTED. | Call Statistics() with small buffer. The return status should be EFI\_BUFFER\_TOO\_SMALL or EFI\_UNSUPPORTED. |
| 5.11.1.8.4 | 0xace9fa20, 0xff34, 0x4fba, 0x8b, 0x95, 0x39, 0xae, 0xca, 0xd9, 0x78, 0x7c | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Statistics - Invokes Statistics() without resetting the statistics and verifying interface correctness returns EFI\_SUCCESSor EFI\_UNSUPPORTED. | Call Statistics() without resetting the statistics and verifying interface correctness. The return status should be EFI\_SUCCESSor EFI\_UNSUPPORTED. |
| 5.11.1.8.5 | 0x3de76704, 0x4bf5, 0x42cd, 0x8c, 0x89, 0x54, 0x7e, 0x4f, 0xad, 0x4f, 0x24 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Statistics - Invokes Statistics(), resetting the statistics, and verifying interface correctness returns EFI\_SUCCESSor EFI\_UNSUPPORTED. | Call Statistics() and reset the statistics and verify interface correctness. The return status should be EFI\_SUCCESSor EFI\_UNSUPPORTED. |

### MCastIPtoMAC()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.9.1 | 0x6880bd92, 0x7004, 0x41b8, 0x9e, 0x43, 0x7b, 0x27, 0x1f, 0xd9, 0xac, 0x2b | EFI\_SIMPLE\_NETWORK\_PROTOCOL.MCastIPtoMAC - Invokes MCastIPtoMAC() when the network interface is not started returns EFI\_NOT\_STARTED. | Call MCastIPtoMAC() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.9.2 | 0x544b08c0, 0x1d26, 0x4462, 0x92, 0x07, 0xdd, 0x7e, 0xb7, 0x54, 0xdc, 0x9e | EFI\_SIMPLE\_NETWORK\_PROTOCOL.MCastIPtoMAC - Invokes MCastIPtoMAC()verifies interface correctness and returns EFI\_SUCCESS. | Call MCastIPtoMAC() and verify interface correctness. The return status should be EFI\_SUCCESS. |

### NvData()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.10.1 | 0x1a0250a2, 0xd085, 0x42ac, 0xb7, 0x42, 0x52, 0x35, 0x26, 0xa1, 0xa9, 0x4f | EFI\_SIMPLE\_NETWORK\_PROTOCOL.NvData - Invokes NvData() when the network interface is not started returns EFI\_NOT\_STARTED. | Call NvData() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.10.2 | 0xd2aaff2b, 0x6632, 0x4d23, 0x98, 0xca, 0x78, 0xd9, 0x0d, 0xea, 0xfb, 0x2f | EFI\_SIMPLE\_NETWORK\_PROTOCOL.NvData - Invokes NvData() with Offset not a multiple of NvRamAccessSize returns EFI\_INVALID\_PARAMETER. | Call NvData() with Offset not a multiple of NvRamAccessSize. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.10.3 | 0xfd0a8da6, 0xe94b, 0x45f0, 0x93, 0x92, 0xe4, 0x8f, 0x9d, 0x09, 0x92, 0xc7 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.NvData - Invokes NvData() with BufferSize not a multiple of NvRamAccessSize returns EFI\_INVALID\_PARAMETER. | Call NvData() with BufferSize not a multiple of NvRamAccessSize. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.10.4 | 0x75fc17ba, 0x5329, 0x4931, 0x96, 0x93, 0xc7, 0x83, 0xf6, 0xac, 0x59, 0xc4 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.NvData - Invokes NvData() with BufferSize + Offset exceeding NvRamSize returns EFI\_INVALID\_PARAMETER. | Call NvData() with BufferSize + Offset exceeding NvRamSize. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.10.5 | 0xba0b2393, 0x0078, 0x434b, 0x99, 0x13, 0xde, 0xa6, 0x6b, 0xdd, 0x83, 0xb3 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.NvData - Invokes NvData() to read (0, n\*NvRamAccessSize) returns EFI\_SUCCESS. | Call NvData() to read (0, n\*NvRamAccessSize) and verify interface correctness. The return status should be EFI\_SUCCESS. |
| 5.11.1.10.6 | 0xf9e2f307, 0x3f73, 0x4c00, 0xbc, 0x31, 0xd5, 0x88, 0xf2, 0x6f, 0x5e, 0xd6 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.NvData - Invokes NvData() to read (NvRamAccessSize, (n‑1)\*NvRamAccessSize) returns EFI\_SUCCESS. | Call NvData() to read (NvRamAccessSize, (n‑1)\*NvRamAccessSize) and verify interface correctness. The return status should be EFI\_SUCCESS. |
| 5.11.1.10.7 | 0x8f18c1d9, 0xbcb2, 0x4e15, 0xaa, 0x16, 0x58, 0xe8, 0x3c, 0x31, 0xd5, 0xe4 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.NvData - Invokes NvData() to read ((n‑1)\*NvRamAccessSize, NvRamAccessSize) returns EFI\_SUCCESS. | Call NvData() to read ((n‑1)\*NvRamAccessSize, NvRamAccessSize) and verify interface correctness. The return status should be EFI\_SUCCESS. |
| 5.11.1.10.8 | 0x443b58d6, 0x683c, 0x4018, 0x89, 0xc9, 0x2e, 0x70, 0xe8, 0x53, 0x6b, 0x7d | EFI\_SIMPLE\_NETWORK\_PROTOCOL.NvData - Invokes NvData() writes and verifies interface correctness, returning EFI\_SUCCESS. | Call NvData() to write and verify interface correctness. The return status should be EFI\_SUCCESS. |

### GetStatus()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.11.1 | 0x21837ad9, 0x942b, 0x4b2b, 0x89, 0x6e, 0xc7, 0xb1, 0xe8, 0xa3, 0x6a, 0xaa | EFI\_SIMPLE\_NETWORK\_PROTOCOL.GetStatus - Invokes GetStatus() when the network interface is not started returns EFI\_NOT\_STARTED. | Call GetStatus() when network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.11.2 | 0xce6f3aba, 0x9d91, 0x4ab4, 0xaa, 0x96, 0x01, 0x14, 0x3e, 0xea, 0xf8, 0x29 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.GetStatus - Invokes GetStatus() when the network interface is not initialized returns EFI\_DEVICE\_ERROR. | Call GetStatus() when the network interface is not initialized. The return status should be EFI\_DEVICE\_ERROR. |
| 5.11.1.11.3 | 0xa1ee7ee5, 0x2b46, 0x4da0, 0xb8, 0x19, 0x0d, 0x10, 0xe1, 0xd0, 0x6f, 0xc0 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.GetStatus - Invokes GetStatus() with an invalid parameter returns EFI\_INVALID\_PARAMETER. | Call GetStatus() when both InterruptStuts and TxBuf are NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.11.4 | 0x8e8f1517, 0x330e, 0x45fd, 0x8d, 0x84, 0x33, 0xff, 0xf1, 0x60, 0x00, 0xf2 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.GetStatus - Invokes GetStatus() verifies interface correctness and returns EFI\_SUCCESS. | Call GetStatus() and verify interface correctness. The return status should be EFI\_SUCCESS. |
| 5.11.1.11.5 | 0xa32b5f48, 0x8215, 0x4024, 0x80, 0x31, 0x33, 0x70, 0x5, 0x20, 0x37, 0x54 | EFI\_SIMPLE\_NETWO RK\_PROTOCOL.GetS tatus - Invokes GetStatus()  to verify the transmitted buffer should be shown up in the recycled transmit buffer. | 1.The transmitted buffer should be shown up in the recycled transmit buffer. |

### Transmit()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.12.1 | 0xfe70e127, 0x6ea1, 0x4ff8, 0xa0, 0x41, 0x1f, 0x96, 0xad, 0x0c, 0xe8, 0x9d | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Invokes Transmit() when the network interface is not started returns EFI\_NOT\_STARTED. | Call Transmit() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.12.2 | 0xfdcadacb, 0x71cd, 0x416c, 0x9a, 0xa6, 0x8c, 0xf5, 0x3a, 0x85, 0x92, 0x05 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Invokes Transmit() when the network interface is not initialized returns EFI\_DEVICE\_ERROR. | Call Transmit() when the network interface is not initialized. The return status should be EFI\_DEVICE\_ERROR. |
| 5.11.1.12.3 | 0xea3773ea, 0x0e0f, 0x45a3, 0x82, 0xa0, 0x64, 0xd4, 0x85, 0xa1, 0x0b, 0x52 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Invokes Transmit() with a HeaderSize value of non‑0 and not equal to MediaHeaderSize returns EFI\_INVALID\_PARAMETER. | Call Transmit() with a HeaderSize value of non‑0 and not equal to MediaHeaderSize. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.12.4 | 0xde544de1, 0x178e, 0x4b5f, 0x97, 0xd7, 0x19, 0x11, 0x9b, 0x1b, 0x7b, 0x18 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Invokes Transmit() with a BufferSize value of less than MediaHeaderSize. | Call Transmit() with a BufferSize value of less than MediaHeaderSize. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.12.5 | 0x4b33c0b2, 0x4ab8, 0x44a0, 0x8c, 0x0b, 0xd9, 0x8b, 0x70, 0x9d, 0xd1, 0x64 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Invokes Transmit() with a Buffer value of NULL returns EFI\_INVALID\_PARAMETER. | Call Transmit() with a Buffer value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.12.6 | 0xa449842c, 0xf5f8, 0x47e9, 0x98, 0x7b, 0x4b, 0x61, 0x41, 0xae, 0xbd, 0x45 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Invokes Transmit() with a HeaderSize value of non‑0 and DestAddr value of NULL returns EFI\_INVALID\_PARAMETER. | Call Transmit() with a HeaderSize value of non‑0 and a DestAddr value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.12.7 | 0x2e3dd087, 0xdd0c, 0x426e, 0x85, 0xba, 0x65, 0xe5, 0x83, 0x10, 0xb1, 0xde | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Invokes Transmit() with a HeaderSize value of non‑0 and a Protocol value of NULL returns EFI\_INVALID\_PARAMETER.. | Call Transmit() with a HeaderSize value of non‑0 and a Protocol value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.12.8 | 0x10e4090b, 0x284b, 0x4886, 0xba, 0x9b, 0x9f, 0x50, 0xc7, 0xff, 0xc5, 0x74 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Invokes Transmit() with a HeaderSize value of non‑0 and a Protocol value of not in accordance with IfType returning EFI\_INVALID\_PARAMETER. | Call Transmit() with a HeaderSize value of non‑0 and a Protocol value of not in accordance with IfType. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.12.9 | 0xdaafbb2a, 0x434b, 0x452f, 0xa6, 0x44, 0xa7, 0x39, 0x2c, 0xf3, 0x59, 0x37 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Calling Transmit()sends Over Sized Packets and returns EFI\_INVALID\_PARAMETER. | Call Transmit() to send Over Sized Packets. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.12.10 | 0x8f8ec6d7, 0x41b5, 0x4e06, 0x87, 0x12, 0xdb, 0x77, 0xba, 0xc6, 0x1a, 0x1f | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Transmit - Calling Transmit() sends Under Sized Packets and returns EFI\_INVALID\_PARAMETER. | Call Transmit() to send Under Sized Packets. The return status should be EFI\_INVALID\_PARAMETER. |

### Receive()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.1.13.1 | 0x6c2503ce, 0x7952, 0x4740, 0x88, 0xd2, 0xe1, 0xb3, 0xa2, 0xd9, 0x5d, 0x2e | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Receive - Invokes Receive() when the network interface is not started returns EFI\_NOT\_STARTED. | Call Receive() when the network interface is not started. The return status should be EFI\_NOT\_STARTED and the state should be “Stopped”. |
| 5.11.1.13.2 | 0xb0def89e, 0xbb48, 0x4829, 0xb5, 0x8e, 0x12, 0x7a, 0xf3, 0x7a, 0x38, 0x9d | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Receive - Invokes Receive() when the network interface is not initialized returns EFI\_DEVICE\_ERROR. | Call Receive() when the network interface is not initialized. The return status should be EFI\_DEVICE\_ERROR. |
| 5.11.1.13.3 | 0xa6783502, 0xf69b, 0x4091, 0xac, 0x09, 0xf0, 0x10, 0x42, 0xa5, 0x93, 0x5e | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Receive - Invokes Receive() with a Buffer value of NULL returns EFI\_INVALID\_PARAMETER. | Call Receive() with a Buffer value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.13.4 | 0xb61dd219, 0x0b04, 0x49b7, 0x9a, 0xf9, 0x8c, 0x5f, 0x27, 0x0c, 0x44, 0x9b | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Receive - Invokes Receive() when BufferSize is smaller than the received Packets returns EFI\_INVALID\_PARAMETER. | Call Receive() when BufferSize is smaller than the received Packets. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.11.1.13.5 | 0x6a319f34, 0x0e40, 0x41aa, 0xae, 0x50, 0x16, 0x9c, 0x4d, 0xe7, 0xb8, 0xc7 | EFI\_SIMPLE\_NETWORK\_PROTOCOL.Receive - Invokes Receive() when no packet is received returns EFI\_NOT\_READY. | Call Receive() when no packet is received. The return status should be EFI\_NOT\_READY. |

## EFI\_PXE\_BASE\_CODE\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_PXE\_BASE\_CODE\_PROTOCOL Section.

### Start()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.2.1.1 | 0x0a483bd1, 0x80cf, 0x463b, 0x8b, 0xb1, 0x2a, 0x33, 0x32, 0x90, 0xcc, 0x08 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.Start - Calling Start() when PXE Protocol is already started returns EFI\_ALREADY\_STARTED. | Call Start() when the EFI\_PXE\_BASE\_CODE\_PROTOCOL is already started. The return code should be EFI\_ALREADY\_STARTED. |
| 5.11.2.1.2 | 0xc1505aee, 0xd73a, 0x416c, 0x9a, 0x3f, 0x9c, 0x00, 0x5d, 0x01, 0xd6, 0xeb | EFI\_PXE\_BASE\_CODE\_PROTOCOL.Start - Calling Start() using IPV6 when PXE Protocol does not support IPV6 returns EFI\_NOT\_SUPPORTED. | Call Start() when EFI\_PXE\_BASE\_CODE\_PROTOCOL does not support IPV6, but require its use. The return code should be EFI\_NOT\_SUPPORTED. |
| 5.11.2.1.3 | 0x13a4a599, 0xb35b, 0x4465, 0xa2, 0xdb, 0xc1, 0xe8, 0xa4, 0xca, 0x9a, 0x93 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.Start – Calling Start() without using IPv6 returns EFI\_SUCCESS. | Call Start() without using IPv6. The return status code should be EFI\_SUCCESS.  Call Start() with using IPv6 if **Ipv6Supported** is **FALSE**. The return status code should be EFI\_UNSUPPORTED. |
| 5.11.2.1.4 | 0x33067ad5, 0xb3a5, 0x44f4, 0x9f, 0xf5, 0xf8, 0x63, 0xda, 0x1f, 0xbd, 0xb3 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.Start – Calling Start() returns correct mode without using IPv6. | Call Start() without using IPv6. The return mode should be correct, including **Started** is TRUE, **UsingIpv6** is FALSE, **AutoArp** is TRUE, and Route Table is **Empty**. |

### Stop()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.2.2.1 | 0x8d75ffa1, 0xdfab, 0x4aff, 0x9f, 0xf7, 0xbb, 0x49, 0x49, 0x08, 0xdc, 0xa3 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.Stop – Calling Stop() while the PXE protocol is already stopped returns EFI\_NOT\_STARTED. | Call Stop() when the EFI\_PXE\_BASE\_CODE\_PROTOCOL is already stopped. The return code should be EFI\_NOT\_STARTED. |
| 5.11.2.2.2 | 0xf88713ff, 0xf149, 0x4e9f, 0x8c, 0xf5, 0x6d, 0x63, 0x55, 0x8f, 0xf2, 0xbd | EFI\_PXE\_BASE\_CODE\_PROTOCOL.Stop – Calling Stop() to disable PXE protocol when it is enabled returns EFI\_SUCCESS. | Enable PXE protocol, and call Stop() to disable PXE protocol. The return code should be EFI\_SUCCESS. |

### Dhcp()

No automatic test is designed to verify this function.

### Discover()

No automatic test is designed to verify this function.

### Mtftp()

No automatic test is designed to verify this function.

### UdpWrite()

No automatic test is designed to verify this function.

### UdpRead()

No automatic test is designed to verify this function.

### SetIpFilter()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.2.8.1 | 0x670cf69f, 0x530a, 0x4bec, 0xaa, 0xb8, 0x41, 0xd3, 0x58, 0x9e, 0x91, 0x99 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.SetIpFilter –Calling SetIpFilter() returns EFI\_SUCCESS. | Enable PXE protocol, and call SetIpFilter(). The returned code should be EFI\_SUCCESS. |
| 5.11.2.8.2 | 0xe9ed28b0, 0x0b88, 0x4e4e, 0xa2, 0xdb, 0xe5, 0xc4, 0xea, 0xd2, 0x00, 0x87 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.SetIpFilter – Calling SetIpFilter() updates IpFilter Mode setting. | Enable PXE protocol, and call SetIpFilter(). The IpFilter filed at EFI\_PXE\_BASE\_CODE\_MODE is updated to the new setting. |
| 5.11.2.8.3 | 0x13317b8d, 0x5d0d, 0x400f, 0x87, 0x4f, 0xaf, 0xe5, 0x08, 0xf1, 0x35, 0x86 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.SetIpFilter – Calling SetIpFilter() with PXE protocol not started returns EFI\_NOT\_STARTED. | Disable PXE protocol, and call SetIpFilter(). The return code should be EFI\_NOT\_STARTED. |

### Arp()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.2.13.1 | 0xdc8b9346, 0xc5c8, 0x4ef5, 0xaf, 0x22, 0xcd, 0xef, 0x81, 0x6d, 0xf6, 0x13 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.Arp – Arp() returns EFI\_INVALID\_PARAMETERwhen IpAddr is NULL | 1.Call Arp()with IpAddr = NULL. The return code must be EFI\_INVALID\_PARAMETER |
| 5.11.2.13.2 | 0xe893562b, 0xcb51, 0x409c, 0xa0, 0x93, 0x7c, 0xad, 0xe1, 0x43, 0xd6, 0xc0 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.Arp – Arp()returns EFI\_UNSUPPORTEDwhen UsingIpv6 is TRUE | 1.Call Arp() when UsingIpv6is TRUE. The return code must be EFI\_UNSUPPORTED |

### SetParameters()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.2.10.1 | 0x3395102a, 0x1b16, 0x4267, 0xb8, 0x5e, 0x88, 0x4b, 0xd6, 0x56, 0xb8, 0x69 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.SetParameters – Calling SetParameters() with PXE protocol not started returns EFI\_NOT\_STARTED. | Disable PXE protocol, and call SetParameters(). The return code should be EFI\_NOT\_STARTED. |

### SetStationIp()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.2.11.1 | 0xe20afad4, 0x04e5, 0x4b09, 0xa2, 0x3a, 0xc0, 0xc1, 0xd5, 0x7f, 0x8b, 0x1b | EFI\_PXE\_BASE\_CODE\_PROTOCOL.SetStationIp – Calling SetStationIp() and modifying IP address and subnet mask returns EFI\_SUCCESS. | Enable PXE protocol, and call SetStationIp() to modify IP address and subnet mask. The returned status code is EFI\_SUCCESS. |
| 5.11.2.11.2 | 0x47feb998, 0x7d0d, 0x4381, 0xae, 0x31, 0x71, 0xbe, 0xdf, 0xb0, 0x73, 0x23 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.SetStationIp – Calling SetStationIp() and only modifying IP address returns EFI\_SUCCESS. | Enable PXE protocol, and call SetStationIp() only to modify IP address. The returned status code is EFI\_SUCCESS. |
| 5.11.2.11.3 | 0x78014f26, 0x0196, 0x4d38, 0xb6, 0xbd, 0x0c, 0x7c, 0x41, 0xf8, 0x5e, 0xa1 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.SetStationIp – Calling SetStationIp() and only modifying subnet mask returns EFI\_SUCCESS. | Enable PXE protocol, and call SetStationIp() only to modify subnet mask of the network device. The returned status code is EFI\_SUCCESS. |
| 5.11.2.11.4 | 0x518491e5, 0xd4ab, 0x42c6, 0x8c, 0x73, 0x90, 0xc1, 0xeb, 0xc2, 0xf1, 0x78 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.SetStationIp – Calling SetStationIp() with PXE not started returns EFI\_NOT\_STARTED. | Disable PXE protocol, and call SetStationIp(). The return code should be EFI\_NOT\_STARTED. |

### SetPackets()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.11.2.12.1 | 0x66c10d09, 0x2578, 0x48b7, 0x80, 0x5b, 0x75, 0xd7, 0x17, 0xcf, 0x71, 0x49 | EFI\_PXE\_BASE\_CODE\_PROTOCOL.SetPackets – Calling SetPackets() with PXE protocol not started returns EFI\_NOT\_STARTED. | Disable PXE protocol, and call SetPackets(). The return code should be EFI\_NOT\_STARTED. |

## EFI\_PXE\_BASE\_CODE\_CALLBACK\_PROTOCOL Test

Reference Document:

*UEFI Specification,* EFI\_PXE\_BASE\_CODE\_CALLBACK\_PROTOCOL Section.

The EFI\_PXE\_BASE\_CODE\_CALLBACK\_PROTOCOL Test is covered in the test for the EFI PXE Base Code Protocol.

## EFI\_BIS\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_BIS\_PROTOCOL Section.

No automatic test is designed to verify this protocol.