# EFI Compliance Test

## EFI Requirements Test

Reference Document:

*UEFI Specification,* Requirements Section.

Configuration

Configuration is a checkpoint in the EFI Requirements Test. If the you need to check the platform-specific protocols, the related profile needs to be updated.

For the correct formatting of profiles, refer to Appendix section A.1, EFI Requirements Test Profile.

### Required Elements

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.22.1.1.1 | 0xf6a871e3, 0xef8a, 0x420f, 0x82, 0x01, 0x35, 0xb6, 0x1c, 0xe2, 0xe8, 0xdb | EFI-Compliant - EFI System Table must be implemented. | 1. The Signature of EFI System Table should be 0x5453595320494249.  2. The Revision of EFI System Table should be equal to or larger than 0x00020000.  3. The Reserved field in EFI System Table should be 0.  4. The RuntimeServices and BootServices pointers of EFI System Table should not be NULL.  5. The CRC32 of EFI System Table must be correct. |
| 5.22.1.1.2 | 0xaddab6ed, 0x5a17, 0x4327, 0x8f, 0xb1, 0x72, 0x93, 0x3d, 0x1a, 0x7b, 0xba | EFI-Compliant - EFI Boot Services Table must be implemented. | 1. The Signature of EFI Boot Services Table should be 0x56524553544f4f42.  2. The Revision of EFI Boot Services Table should be equal to or larger than 0x00020000.  3. The Reserved field in EFI Boot Services Table should be 0.  4. No function pointers in EFI Boot Services Table should be NULL. |
| 5.22.1.1.3 | 0x13a20958, 0xc860, 0x452f, 0xb9, 0xa2, 0xe6, 0xd9, 0x96, 0x41, 0x92, 0x24 | EFI-Compliant - EFI Runtime Services Table must be implemented. | 1. The Signature of EFI Runtime Services Table should be 0x56524553544e5552.  2. The Revision of EFI Runtime Services Table should be equal to or larger than 0x00020000.  3. The Reserved field in EFI Runtime Services Table should be 0.  4. No function pointers in EFI Runtime Services Table should be NULL. |
| 5.22.1.1.4 | 0xa82f8d56, 0x1476, 0x41f1, 0xba, 0xc4, 0x97, 0x59, 0x79, 0x9f, 0x97, 0xf3 | EFI-Compliant –EFI\_LOADED\_IMAGE\_PROTOCOL must exist. | 1. Call LocateProtocol() to find the LOADED\_IMAGE\_PROTOCOL. The return status should be EFI\_SUCCESS. |
| 5.22.1.1.5 | 0xf61f0f0a, 0x64fe, 0x40a6, 0x9d, 0x7c, 0x07, 0x46, 0xa2, 0x30, 0x24, 0x5f | EFI-Compliant –EFI\_DEVICE\_PATH\_PROTOCOL must exist. | 1. Call LocateProtocol() to find the DEVICE\_PATH\_PROTOCOL. The return status should be EFI\_SUCCESS. |
| 5.22.1.1.6 | 0x02c017d7, 0x1557, 0x47d9, 0xbc, 0xe9, 0x87, 0x18, 0x2d, 0x07, 0x91, 0x0c | EFI-Compliant –EFI\_DECOMPRESS\_PROTOCOL must exist. | 1. Call LocateProtocol() to find the DECOMPRESS\_PROTOCOL. The return status should be EFI\_SUCCESS.  2. No function pointers in DECOMPRESS\_PROTOCOL should be NULL. |
| 5.22.1.1.7 | 0x3a07dc1b, 0x53d1, 0x4fac, 0x88, 0xaf, 0xc7, 0x25, 0x79, 0xeb, 0x07, 0xf2 | UEFI-Compliant-EFI\_DEVICE\_PATH\_UTILITIES\_PROTOCOL must exist | 1. Call LocateProtocol() to find the EFI\_DEVICE\_PATH\_UTILITIES\_PROTOCOL, the return status should be EFI\_SUCCESS  2. No function pointer in Device Path Utility protocol should be NULL |
| 5.22.1.1.8 | 0xf6334f9b, 0xb930, 0x4adb, 0xa5, 0x3b, 0x76, 0xfa, 0x7b, 0x4c, 0x27, 0x62 | UEFI-Compliant  The EFI\_GLOBAL\_VARIABLE guid should be used by the globally defined variables only, and the attributes of the variables should be same with the definition in the Specification. | 1. Locate all variables with EFI\_GLOBAL\_VARIABLE guid, check the variable name is in the pre-defined globally variable list.  2. Check the variable attribute. |

### Platform-Specific Elements

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.22.1.2.1 | 0x8f7556c2, 0x4665, 0x4353, 0xa3, 0xaf, 0x9c, 0x00, 0x5a, 0x1e, 0x63, 0xe1 | EFI-Compliant - EFI\_SIMPLE\_ TEXT\_INPUT\_PROTOCOL, EFI\_SIMPLE\_ TEXT\_INPUT\_EX\_PROTOCOL and EFI\_SIMPLE\_TEXT\_OUT\_PROTOCOL must be implemented if a platform includes console devices. | 1. Call LocateProtocol() to find the EFI\_SIMPLE\_ TEXT\_INPUT\_PROTOCOL.  2. Call LocateProtocol() to find the EFI\_SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.  3. Call LocateProtocol() to find the EFI\_SIMPLE\_TEXT\_OUT\_PROTOCOL.  4. If the INI file indicates that the platform includes console devices, the return status in steps 1, 2 and 3 should be EFI\_SUCCESS. If not, the return status in steps 1, 2 and 3 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.2 | 0x72ba0e86, 0x58e5, 0x48dd, 0x85, 0x29, 0x88, 0xc6, 0x83, 0x83, 0x11, 0x8d | UEFI-Compliant - EFI\_GRAPHICS\_OUTPUT\_PROTOCOL, EFI\_EDID\_ACTIVE\_PROTOCOL, EFI\_EDID\_DISCOVERED\_PROTOCOL must be implemented if a platform includes graphical console devices. | 1. Call LocateProtocol() to find the EFI\_GRAPHICS\_OUTPUT\_PROTOCOL  2. Call LocateProtocol() to find the EFI\_EDID\_ACTIVE\_PROTOCOL. ,  3. Call LocateProtocol() to find the EFI\_EDID\_DISCOVERED\_PROTOCOL.  4. If the INI file indicates that the platform includes graphical console devices, the return status in all steps 1, 2 and 3 should be EFI\_SUCCESS.  5. If the INI file doesn’t indicate that the platform includes graphical console devices, the return status in all steps 1, 2 and 3 could be either EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.3 | 0x18670db1, 0x89fb, 0x4de4, 0xb1, 0x0f, 0x89, 0x8e, 0x04, 0x7d, 0x95, 0x2a | UEFI-Compliant –EFI\_SIMPLE\_POINTER\_PROTOCOL must be implemented if a platform includes a pointer device as part of its console support. | 1. Call LocateProtocol() to find the EFI\_SIMPLE\_POINTER\_PROTOCOL.  2. If the INI file indicates that the platform includes a pointer device, the return status in step 1 should be EFI\_SUCCESS.  3. If the INI file doesn’t indicate that the platform includes a pointer device, the return status in step 1 could be either EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.4 | 0xbf38a3fd, 0x58ac, 0x419a, 0xab, 0xc2, 0xc6, 0x0b, 0xae, 0x9c, 0xfe, 0x67 | UEFI-Compliant –EFI\_BLOCK\_IO\_PROTOCOL, EFI\_DISK\_IO\_PROTOCOL, EFI\_SIMPLE\_FILE\_SYSTEM; EFI\_UNICODE\_COLLATION\_PROTOCOL must be implemented if a platform supports booting from a disk. | 1. Call LocateProtocol() to find the EFI\_BLOCK\_IO\_PROTOCOL protocol.  2. Call LocateProtocol() to find the EFI\_DISK\_IO\_PROTOCOL.  3. Call LocateProtocol() to find the EFI\_SIMPLE\_FILE\_SYSTEM\_PROTOCOL.  4. Call LocateProtocol() to find the EFI\_UNICODE\_COLLATION\_PROTOCOL.  5. If the INI file indicates that the platform supports booting from a disk, the return status in steps 1, 2, 3, and 4 all should be EFI\_SUCCESS.  6. If the INI file doesn’t indicate that the platform supports booting from a disk, the return status in steps 1, 2, 3, and 4 all should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.5 | 0x98551ae7, 0x5020, 0x4ddd, 0x86, 0x1a, 0xcf, 0xff, 0xb4, 0xd6, 0x03, 0x82 | UEFI-Compliant – EFI\_PXE\_BASE\_CODE\_PROTOCOL must be implemented if a platform supports TFTP-based booting from a network device. And platform must be prepared to produce this protocol on any of EFI\_NETWORK\_INTERFACE\_IDENTIFIER\_PROTOCOL (UNDI),  **EFI\_SIMPLE\_NETWORK\_PROTOCOL**, or the **EFI\_MANAGED\_NETWORK\_PROTOCOL**. If platform supports validating the image received from network device, **SetupMode** equal zero. | 1. Call **LocateProtocol()** to find the **EFI\_PXE\_BASE\_CODE\_PROTOCOL**. 2. Call **LocateProtocol()** to find the **EFI\_NETWORK\_INTERFACE\_IDENTIFIER\_PROTOCOL**, **EFI\_SIMPLE\_NETWORK\_PROTOCOL**, **EFI\_MANAGED\_NETWORK\_PROTOCOL**. 3. If the INI file indicates that the platform supports TFTP-based booting from a network device, the return status in step 1 should be **EFI\_SUCCESS**. And one of the step 2 should be **EFI\_SUCCESS** at least. 4. If the INI file doesn’t indicate that the platform supports TFTP-based booting from a network device, the return status in both step 1 and step 2 step should be **EFI\_SUCCESS** or **EFI\_ERROR**. 5. If the INI file indicates that the platform supports validating the image received from a network device, **SetupMode** equal zero. |
| 5.22.1.2.6 | 0x517bcbeb, 0x4982, 0x4a7e, 0x85, 0x51, 0xca, 0x84, 0x7d, 0xdc, 0x21, 0xc2 | UEFI-Compliant –EFI\_SERIAL\_IO\_PROTOCOL must be implemented if a platform includes a byte stream device. | 1. Call LocateProtocol() to find the EFI\_SERIAL\_IO\_PROTOCOL.  2. If the INI file indicates that the platform includes a byte-stream device, the return status in step 1 should be EFI\_SUCCESS.  3. If the INI file doesn’t indicate that the platform includes a byte-stream device, the return status in step 1 step could be either EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.7 | 0x213a75c9, 0x7f3d, 0x42db, 0xb3, 0x2a, 0x02, 0xdb, 0xd6, 0x98, 0x31, 0x9d | UEFI-Compliant –EFI\_PCI\_ROOT\_BRIDGE\_IO\_PROTOCOL and EFI\_PCI\_IO\_PROTOCOL must be implemented if a platform includes PCI bus support. | 1. Call LocateProtocol() to find the EFI\_PCI\_ROOT\_BRIDGE\_IO \_PROTOCOL.  2. Call LocateProtocol() to find the EFI\_PCI\_IO\_PROTOCOL.  3. If the INI file indicates that the platform includes PCI bus support, the return status in both steps 1 and 2 should be EFI\_SUCCESS.  4. If the INI file doesn’t indicate that the platform includes PCI bus support, the return status in both steps 1 and 2 steps could be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.8 | 0x0ccd5843, 0x5bb5, 0x4fc2, 0xa7, 0x32, 0xdb, 0x17, 0xc4, 0x14, 0xa4, 0x3d | UEFI-Compliant –EFI\_USB\_HC2\_PROTOCOL and EFI\_USB\_IO\_PROTOCOL must be implemented if a platform includes USB bus support. | 1. Call LocateProtocol() to find the EFI\_USB\_HC2\_PROTOCOL.  2. Call LocateProtocol() to find the EFI\_USB\_IO\_PROTOCOL.  3. If INI file indicates the platform includes USB bus support, the return status in 1 and 2 steps should be both EFI\_SUCCESS.  4. If INI file doesn’t indicate the platform includes USB bus support, the return status in 1 and 2 steps should be both EFI\_SUCCESS or both EFI\_ERROR. |
| 5.22.1.2.9 | 0x2b83418f, 0xe7fb, 0x4528, 0xb6, 0xff, 0xc9, 0xd4, 0x87, 0xae, 0x2e, 0xff | UEFI-Compliant –EFI\_EXT\_SCSI\_PASS\_THRU\_PROTOCOL must be implemented if a platform includes an I/O system that uses SCSI command packets. | 1. Call LocateProtocol() to find the EFI\_EXT\_SCSI\_PASS\_THRU\_PROTOCOL.  2. If INI file indicates the platform includes an I/O system that uses SCSI command packets, the return status in 1 step should be EFI\_SUCCESS.  3. If INI file doesn’t indicate the platform includes an I/O system that uses SCSI command packets, the return status in 1 step could be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.10 | 0x3ee22696, 0x0875, 0x46f4, 0x88, 0x84, 0xba, 0x12, 0x4c, 0x7e, 0xaf, 0xf0 | UEFI-Compliant –EFI\_DEBUG\_SUPPORT\_PROTOCOL and EFI\_DEBUG\_PORT\_PROTOCOL must be implemented if a platform supports debugging capabilities. | 1. Call LocateProtocol() to find the EFI\_DEBUG\_SUPPORT\_PROTOCOL.  2. Call LocateProtocol() to find the EFI\_DEBUG\_PORT\_PROTOCOL.  3. If INI file indicates the platform supports debugging capabilities, the return status in 1 and 2 steps should be both EFI\_SUCCESS.  4. If INI file doesn’t indicate the platform supports debugging capabilities, the return status in 1 and 2 steps should be both EFI\_SUCCESS or both EFI\_ERROR. |
| 5.22.1.2.11 | 0x329027ce, 0x406e, 0x48c8, 0x8a, 0xc1, 0xa0, 0x2c, 0x1a, 0x6e, 0x39, 0x83 | UEFI-Compliant –EFI\_PLATFORM\_DRIVER\_OVERRIDE\_PROTOCOL must be implemented if a platform includes the ability to override the default driver. | 1. Call LocateProtocol() to find the EFI\_PLATFORM\_DRIVER\_OVERRIDE\_PROTOCOL.  2. If INI file indicates the platform includes the ability to override the default driver, the return status in 1 step should be EFI\_SUCCESS.  3. If INI file doesn’t indicate the platform includes the ability to override the default driver, the return status in 1 step could be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.12 | 0x76a6a1b0, 0x8c53, 0x407d, 0x84, 0x86, 0x9a, 0x6e, 0x63, 0x32, 0xd3, 0xce | UEFI-Compliant – **EFI\_MANAGED\_NETWOR K\_PROTOCOL**, **EFI\_MANAGED\_NETWOR K\_SERVICE\_BINDING\_ PROTOCOL, EFI\_ARP\_PROTOCOL, EFI\_ARP\_SERVICE\_BI NDING\_PROTOCOL, EFI\_DHCP4\_PROTOCOL , EFI\_DHCP4\_SERVICE\_ BINDING\_PROTOCOL, EFI\_TCP4\_PROTOCOL, EFI\_TCP4\_SERVICE\_B INDING\_PROTOCOL, EFI\_IP4\_PROTOCOL, EFI\_IP4\_SERVICE\_BI NDING\_PROTOCOL, EFI\_IP4\_CONFIG2\_PRO TOCOL, EFI\_UDP4\_PROTOCOL, EFI\_UDP4\_SERVICE\_B INDING\_PROTOCOL, EFI\_MTFTP4\_PROTOCO L,** and **EFI\_MTFTP4\_SERVICE \_BINIING\_PROTOCOL** are required for general network application | 1. Call **LocateProtocol()** to find the **EFI\_MANAGED\_NETWORK\_PROTOCOL**, **EFI\_MANAGED\_NETWORK\_SERVICE\_BINDING\_PROTOCOL,** **EFI\_ARP\_PROTOCOL, EFI\_ARP\_SERVICE\_BINDING\_PROTOCOL, EFI\_DHCP4\_PROTOCOL, EFI\_DHCP4\_SERVICE\_BINDING\_PROT OCOL, EFI\_TCP4\_PROTOCOL, EFI\_TCP4\_SERVICE\_BINDING\_PROTO COL, EFI\_IP4\_PROTOCOL, EFI\_IP4\_SERVICE\_BINDING\_PROTOC OL, EFI\_IP4\_CONFIG2\_PROTOCOL, EFI\_UDP4\_PROTOCOL, EFI\_UDP4\_SERVICE\_BINDING\_PROTO COL, EFI\_MTFTP4\_PROTOCOL,** and **EFI\_MTFTP4\_SERVICE\_BINIING\_PRO TOCOL** 2. If INI file indicates the platform includes the ability to general network application, the return status for locating all protocols described in step 1 should be **EFI\_SUCCESS** 3. If INI file doesn’t indicate the platform includes the ability for general network application, the return status for locating all protocols described in step 1 could |
| 5.22.1.2.13 | 0x28c068f2, 0xf398, 0x488a, 0xb0, 0x59, 0x53, 0x4e, 0x98, 0x2d, 0x9c, 0x85 | UEFI-Compliant –EFI\_SCSI IO\_PROTOCOL, EFI\_Block IO\_PROTOCOL and EFI\_EXT\_SCSI\_PASS\_THRU\_PROTOCOL must be implemented if a platform supports booting from a SCSI peripheral device. | 1. Call LocateProtocol() to find the EFI\_SCSI IO\_PROTOCOL.  2. Call LocateProtocol() to find the EFI\_Block IO\_PROTOCOL protocol.  3. Call LocateProtocol() to find the EFI\_EXT\_SCSI\_PASS\_THRU\_PROTOCOL.  4. If the INI file indicates that the platform supports booting from a network device, the return status in all steps 1, 2 and 3 should be EFI\_SUCCESS.  5. If the INI file doesn’t indicate that the platform supports booting from a network device, the return status in all steps 1, 2 and 3 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.14 | 0x6b7077a6, 0x4b13, 0x4e13, 0x9b, 0x1f, 0x0c, 0x4b, 0x3a, 0x86, 0x69, 0xe2 | UEFI-Compliant – **EFI\_ISCSI\_INITIATOR\_NAME\_PROTOCOL** and **EFI\_AUTHENTICATION\_INFO\_PROTOCOL** must be implemented if a platform supports booting from a ISCSI peripheral device. | 1. Call **LocateProtocol()** to find the **EFI\_ISCSI\_INITIATOR\_NAME\_PROTO COL** and **EFI\_AUTHENTICATION\_INFO\_PROTOCOL**. 2. If the INI file indicates that the platform supports booting from a iSCSI peripheral, the return status in both steps 1 should be **EFI\_SUCCESS**. 3. If the INI file doesn’t indicate that the platform supports booting from iSCSI peripheral, the return status in steps 1 should be **EFI\_SUCCESS** or **EFI\_ERROR**. |
| 5.22.1.2.15 | 0x4c82eb2d, 0xc785, 0x410c, 0x95, 0xd1, 0xae, 0x27, 0x12, 0x21, 0x44, 0xc8 | UEFI Compliant –UEFI V6 General Network Driver Dhcp6SB, Tcp6SB, Ip6SB, Udp6SB, Ip6Config, Vlan must exist if a platform supports V6 network stack | 1. Call LocateProtocol() to find the V6 network stack.  2. If the INI file indicates that the platform supports v6 stack  , the return status in step 1 should be EFI\_SUCCESS.  3. If the INI file doesn’t indicate that the platform supports v6 network stack, the return status in steps1 should be EFI\_SUCCESSor EFI\_ERROR. |
| 5.22.1.2.16 | 0x1d0a2f2a, 0x924, 0x4b8c, 0x9f, 0xc7, 0xb1, 0x85, 0xcc, 0x22, 0xe1, 0x18 | UEFI Compliant –UEFI EBC interpreter must exist if a platform supports EBC image | 1. Call **LocateProtocol()** to find the **EFI\_EBC\_PROTOCOL**.  2. If the INI file indicates that the platform supports EBC image  , the return status in step 1 should be **EFI\_SUCCESS**.  3. If the INI file doesn’t indicate that the platform supports EBC image, the return status in step 1 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.17 | 0xb7cd2d76, 0xea43, 0x4013, 0xb7, 0xd1, 0x59, 0xeb, 0x2e, 0xc9,  0xbf, 0x1b | UEFI Compliant –UEFI HiiDatabase, HiiString, HiiConfigRouting, HiiConfigAccess must be existed if the platform supports HII. If it supports bitmapped fonts, then HiiFont must exist also. | 1. Call LocateProtocol() to find HiiDatabase, HiiString, HiiConfigRouting, HiiConfigAccess.  2. If the INI file indicates that the platform supports HII all return statuses in step 1 should be EFI\_SUCCESS.  3. If the INI file doesn’t indicate that the platform supports HII, the return status in step1 should be EFI\_SUCCESS or EFI\_ERROR.  4. If step 2 is true, and the INI file indicates the platform support bitmapped font, call LocateProtocol() to find HiiFont, and the return status should EFI\_SUCCESS. |
| 5.22.1.2.18 | 0x5aea7246, 0xbcf9,  0x4ba4, 0x81, 0xd2, 0x83, 0x2c, 0x98, 0x41, 0x46, 0xf3 | UEFI-Compliant –  EFI\_NVM\_EXPRESS \_PASS\_THRU\_PROTOCOL must be implemented if a platform includes an NVM Express controller | 1. Call LocateProtocol() to find the EFI\_NVM\_EXPRESS\_PASS\_THRU\_PROTOCOL. 2. If the INI file indicates that the platform  includes an NVM Express controller, the return status in steps 1 should be EFI\_SUCCESS. 3. If the INI file doesn’t indicate that the platform includes an NVM Express controller, the return status in steps 1 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.19 | 0x5cb0cdb5, 0xac80, 0x4983, 0xb7, 0x10, 0x4b, 0xb, 0xf0, 0x19, 0x15, 0x63 | UEFI Compliant –  EFI\_BLOCK\_IO\_PROTOCOL must be existed if the platform supports booting from a block-oriented NVM Express controller. EFI\_NVM\_EXPRESS\_PASS\_THRU\_PROTOCOL may be required. | 1. Call LocateProtocol() to find EFI\_BLOCK\_IO\_PROTOCOL. 2. If the INI file indicates that the platform supports  booting from a block-oriented NVM Express controller, all return statuses in step 1 should be EFI\_SUCCESS. 3. If the INI file doesn’t indicate that the platform supports booting from a block-oriented NVM Express controller, the return status in step1 should be EFI\_SUCCESS or EFI\_ERROR. 4. If step 2 is true, and the INI file indicates the platform support EFI\_NVM\_EXPRESS\_PASS\_THRU\_PROTOCOL, call LocateProtocol() to find it, and the return status should be EFI\_SUCCESS. |
| 5.22.1.2.20 | 0x563f654f, 0xaba8, 0x4539, 0x80, 0x4b, 0x50, 0x63, 0x5, 0x7, 0x26, 0x23 | UEFI-Compliant –  EFI\_ATA\_PASS\_THRU\_PROTOCOL must be implemented if a platform  includes an I/O subsystem that utilizes ATA command packets. | 1. Call LocateProtocol() to find the EFI\_ATA\_PASS\_THRU\_PROTOCOL. 2. If the INI file indicates that the platform includes an I/O subsystem that utilizes ATA command packets, the return status in steps 1 should be EFI\_SUCCESS. 3. If the INI file doesn’t indicate that the platform includes an I/O subsystem that utilizes ATA command packets, the return status in steps 1 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.21 | 0x2e6d1733, 0x6d39, 0x49ab, 0xa8, 0x86, 0x1b, 0x6d, 0xe4, 0x45, 0x66, 0xa8 | UEFI Compliant –  EFI\_DNS4\_PROTOCOL, EFI\_DNS4\_SERVICE\_BINDING\_PROTOCOL must be existed if the platform supports DNS for IPv4 stack. | 1. Call LocateProtocol() to find the EFI\_DNS4\_PROTOCOL and EFI\_DNS4\_SERVICE\_BINDING\_PROTOCOL. 2. If the INI file indicates that the platform supports DNS for IPv4 stack, the return status in steps 1 should be EFI\_SUCCESS. 3. If the INI file doesn’t indicate this capability, the return status in steps 1 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.22 | 0xe02a6ef3, 0x4b70, 0x40ec, 0xaa, 0x23, 0x50, 0xb7, 0xb9, 0x72, 0xb0, 0x65 | UEFI Compliant –  EFI\_DNS6\_PROTOCOL, EFI\_DNS6\_SERVICE\_BINDING\_PROTOCOL must be existed if the platform supports DNS for IPv6 stack. | 1. Call LocateProtocol() to find the EFI\_DNS6\_PROTOCOL and EFI\_DNS6\_SERVICE\_BINDING\_PROTOCOL. 2. If the INI file indicates that the platform supports DNS for IPv6 stack, the return status in steps 1 should be EFI\_SUCCESS. 3. If the INI file doesn’t indicate this capability, the return status in steps 1 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.23 | 0xcb6f7b77, 0xb15, 0x43f7, 0xa9, 0x5b, 0x8c, 0x7f, 0x9f, 0xd7, 0xb, 0x21 | UEFI Compliant – EFI\_TLS\_PROTOCOL, EFI\_TLS\_SERVICE\_BINDING\_PROTOCOL, EFI\_TLS\_CONFIGURATION\_PROTOCOL must be existed if the platform supports TLS feature. | 1. Call LocateProtocol() to find the EFI\_TLS\_PROTOCOL, EFI\_TLS\_SERVICE\_BINDING\_PROTOCOL and EFI\_TLS\_CONFIGURATION\_PROTOCOL. 2. If the INI file indicates that the platform supports TLS feature, the return status in steps 1 should be EFI\_SUCCESS. 3. If the INI file doesn’t indicate this capability, the return status in steps 1 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.24 | 0x77fddb95, 0x5969, 0x4fb4, 0xa2, 0x18, 0x5c, 0xc, 0x76, 0xb, 0x5, 0x64 | UEFI Compliant –  EFI\_HTTP\_PROTOCOL, EFI\_HTTP\_SERVICE\_BINDING\_PROTOCOL, EFI\_HTTP\_UTILITIES\_PROTOCOL must be existed if the platform includes the ability to perform a HTTP-based boot from a network device. | 1. Call LocateProtocol() to find the EFI\_HTTP\_PROTOCOL, EFI\_HTTP\_SERVICE\_BINDING\_PROTOCOL and EFI\_HTTP\_ UTILITIES\_PROTOCOL. 2. If the INI file indicates that the platform includes the ability to perform a HTTP-based boot from a network device, the return status in steps 1 should be EFI\_SUCCESS. 3. If the INI file doesn’t indicate this capability, the return status in steps 1 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.25 | 0xf0dc12  fa, 0x3c4b, 0x43f7, 0xa6, 0x9e, 0xa5, 0xbe, 0x6f, 0xcc, 0x90, 0xa1 | UEFI Compliant –  EFI\_EAP\_PROTOCOL, EFI\_EAP\_CONFIGURATION\_PROTOCOL, EFI\_EAP\_MANAGEMENT2\_PROTOCOL must be existed if the platform includes the ability to perform a wireless boot from a network device with EAP feature, and if this platform provides a standalone wireless EAP driver. | 1. Call LocateProtocol() to find the EFI\_EAP\_PROTOCOL, EFI\_EAP\_CONFIGURATION\_PROTOCOL and EFI\_EAP\_MANAGEMENT2\_PROTOCOL. 2. If the INI file indicates that the platform includes the ability to perform a wireless boot from a network device with EAP feature, and if this platform provides a standalone wireless EAP driver, the return status in steps 1 should be EFI\_SUCCESS. 3. If the INI file doesn’t indicate this capability, the return status in steps 1 should be EFI\_SUCCESS or EFI\_ERROR. |
| 5.22.1.2.26 | 0x87e50392, 0xf5a2, 0x42b8, 0x81, 0x12, 0x68, 0xbe, 0xc9, 0x2, 0xb9, 0xbc | UEFI Compliant –  EFI\_BLUETOOTH\_HC\_PROTOCOL, EFI\_  BLUETOOTH\_IO\_PROTOCOL, EFI\_BLUETOOTH\_CONFIG\_PROTOCOL must be existed if the platform supports classic Bluetooth. | 1. Call LocateProtocol() to find the UEFI Compliant –UEFI EFI\_BLUETOOTH\_HC\_PROTOCOL, EFI\_BLUETOOTH\_IO\_PROTOCOL and EFI\_BLUETOOTH\_CONFIG\_PROTOCOL. 2. If the INI file indicates that the platform supports classic Bluetooth, the return status in steps 1 should be EFI\_SUCCESS. 3. If the INI file doesn’t indicate this capability, the return status in steps 1 should be EFI\_SUCCESS or EFI\_ERROR. |