* Deprecated Protocols

This appendix lists the Protocol , GUID, and revision identifier name changes and the deprecated protocols compared to the *EFI Specification 1.10.* The protocols listed are not Runtime, Reentrant or MP Safe. Protocols are listed by EFI 1.10 name.

For protocols in the table whose TPL is not <= TPL\_NOTIFY:

This function must be called at a TPL level less then or equal to %%%%.

%%%% is TPL\_CALLBACK or TPL\_APPLICATION. The <= is done via text.

* Protocol Name changes

|  |  |
| --- | --- |
| EFI 11.0 Protocol Name | UEFI 2.0 Protocol Name |
| EFI\_LOADED\_IMAGE | EFI\_LOADED\_IMAGE\_PROTOCOL |
| TPL | <= TPL\_NOTIFY |
| New GUID name | EFI\_LOADED\_IMAGE\_PROTOCOL\_GUID |
| EFI\_DEVICE\_PATH | EFI\_DEVICE\_PATH\_PROTOCOL |
| TPL | <= TPL\_NOTIFY |
| New GUID name | EFI\_DEVICE\_PATH\_PROTOCOL\_GUID |
| SIMPLE\_INPUT\_INTERFACE | EFI\_SIMPLE\_INPUT\_PROTOCOL |
| TPL | <= TPL\_APPLICATION |
| New GUID name | EFI\_SIMPLE\_INPUT\_PROTOCOL\_GUID |
| SIMPLE\_TEXT\_OUTPUT\_INTERFACE | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL |
| TPL | <=TPL\_CALLBACK |
| New GUID name | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL\_GUID |
| SERIAL\_IO\_INTERFACE | EFI\_SERIAL\_IO\_PROTOCOL |
| TPL | <=TPL\_CALLBACK |
| New GUID name | EFI\_SERIAL\_IO\_PROTOCOL\_GUID |
| EFI\_LOAD\_FILE\_INTERFACE | EFI\_LOAD\_FILE\_PROTOCOL |
| TPL | <= TPL\_NOTIFY |
| New GUID name | EFI\_LOAD\_FILE\_PROTOCOL\_GUID |
| EFI\_FILE\_IO\_INTERFACE | EFI\_SIMPLE\_FILE\_SYSTEM\_PROTOCOL |
| TPL | <=TPL\_CALLBACK |
| New GUID name | EFI\_FILE\_SYSTEM\_PROTOCOL\_GUID |
| EFI\_FILE | EFI\_FILE\_PROTOCOL |
| TPL | <= TPL\_CALLBACK |
| New GUID name | EFI\_FILE\_PROTOCOL\_GUID |
| EFI\_DISK\_IO | EFI\_DISK\_IO\_PROTOCOL |
| TPL | <=TPL\_CALLBACK |
| New GUID name | EFI\_DISK\_IO\_PROTOCOL\_GUID |
| EFI\_BLOCK\_IO | EFI\_BLOCK\_IO\_PROTOCOL |
| TPL | <=TPL\_CALLBACK |
| New GUID name | EFI\_BLOCK\_IO\_PROTOCOL\_GUID |
| UNICODE\_COLLATION\_INTERFACE | EFI\_UNICODE\_COLLATION\_PROTOCOL |
| TPL | <= TPL\_NOTIFY |
| New GUID name | EFI\_UNICODE\_COLLATION\_PROTOCOL\_GUID |
| EFI\_SIMPLE\_NETWORK | EFI\_SIMPLE\_NETWORK\_PROTOCOL |
| TPL | <=TPL\_CALLBACK |
| New GUID name | EFI\_SIMPLE\_NETWORK\_PROTOCOL\_GUID |
| EFI\_NETWORK\_INTERFACE\_IDENTIFIER  \_INTERFACE | EFI\_NETWORK\_INTERFACE\_IDENTIFIER\_PROTOCOL |
| TPL | <= TPL\_NOTIFY |
| New GUID name | EFI\_NETWORK\_INTERFACE\_IDENTIFIER\_PROTOCOL\_GUID |
| EFI\_PXE\_BASE\_CODE | EFI\_PXE\_BASE\_CODE\_PROTOCOL |
| TPL | <= TPL\_NOTIFY |
| New GUID name | EFI\_ PXE\_BASE\_CODE \_PROTOCOL\_GUID |
| EFI\_PXE\_BASE\_CODE\_CALLBACK | EFI\_PXE\_BASE\_CODE\_CALLBACK\_PROTOCOL |
| TPL | <= TPL\_NOTIFY |
| New GUID name | EFI\_PXE\_BASE\_CODE\_CALLBACK\_PROTOCOL\_GUID |
| EFI\_DEVICE\_IO\_INTERFACE | EFI\_DEVICE\_IO\_PROTOCOL |
| TPL | <= TPL\_NOTIFY |
| New GUID name | EFI\_DEVICE\_IO\_PROTOCOL\_GUID |

* Revision Identifier Name Changes

|  |  |
| --- | --- |
| EFI 11.0 Revision Identifier Name | UEFI 2.0 Revision Identifier Name |
| EFI\_LOADED\_IMAGE\_INFORMATION\_REVISION | EFI\_LOADED\_IMAGE\_PROTOCOL\_REVISION |
| SERIAL\_IO\_INTERFACE\_REVISION | EFI\_SERIAL\_IO\_PROTOCOL\_REVISION |
| EFI\_FILE\_IO\_INTERFACE\_REVISION | EFI\_SIMPLE\_FILE\_SYSTEM\_PROTOCOL\_REVISION |
| EFI\_FILE\_REVISION | EFI\_FILE\_PROTOCOL\_REVISION |
| EFI\_DISK\_IO\_INTERFACE\_REVISION | EFI\_DISK\_IO\_PROTOCOL\_REVISION |
| EFI\_BLOCK\_IO\_INTERFACE\_REVISION | EFI\_BLOCK\_IO\_PROTOCOL\_REVISION |
| EFI\_SIMPLE\_NETWORK\_INTERFACE\_REVISION | EFI\_SIMPLE\_NETWORK\_PROTOCOL\_REVISION |
| EFI\_NETWORK\_INTERFACE\_IDENTIFIER\_INTERFACE  \_REVISION | EFI\_NETWORK\_INTERFACE\_IDENTIFIER\_PROTOCOL  \_REVISION |
| EFI\_PXE\_BASE\_CODE\_INTERFACE\_REVISION | EFI\_PXE\_BASE\_CODE\_PROTOCOL\_REVISION |
| EFI\_PXE\_BASE\_CODE\_CALLBACK\_INTERFACE  \_REVISION | EFI\_PXE\_BASE\_CODE\_CALLBACK\_PROTOCOL  \_REVISION |

* Deprecated Protocols

Device I/O Protocol

The support of the Device I/O Protocol (see EFI 1.1 Chapter 18) has been replaced by the use of the **PCI Root Bridge I/O** protocols from the UEFI 2.0 specification and following. Note: certain “legacy” EFI applications such as some of the ones that reside in the EFI Toolkit assume the presence of Device I/O.

 UGA I/O + UGA Draw Protocol

The support of the UGA \* Protocols (see EFI 1.1 Section 10.7) have been replaced by the use of the **EFI Graphics Output Protocol** described in the UEFI 2.0 specification.

USB Host Controller Protocol (version that existed for EFI 1.1)

The support of the USB Host Controller Protocol (see EFI 1.1 Section 14.1) has been replaced by the use of a UEFI 2.0 instance that covers both USB 1.1 and USB 2.0 support, as described in the UEFI 2.0 specification and following. It replaces the pre-existing protocol definition.

SCSI Passthru Protocol

The support of the SCSI Passthru Protocol (see EFI 1.1 Section 13.1) has been replaced by the use of the **Extended SCSI Passthru Protocol** which is described in the UEFI 2.0 specification.

BIS Protocol

Remains as an optional protocol.

* �EFI\_UGA\_DRAW\_PROTOCOL Test

Reference Document:

*Specification*, Section .

* GetMode()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.3.1.1 | 0x7be3c5ea, 0xca81, 0x49e2, 0xba, 0xc6, 0xb9, 0xa6, 0x5b, 0xbf, 0xfc, 0x57 | EFI\_UGA\_DRAW\_PROTOCOL.GetMode – GetMode() with valid parameter returns EFI\_SUCCESS | 1. Call GetMode() with valid parameter to backup current UGA mode. The return code should be EFI\_SUCCESS |
| 5.6.3.1.2 | 0x2dcf2f9d, 0xbc9c, 0x4be2, 0x9d, 0x0a, 0x35, 0xb9, 0x9d, 0x13, 0xb1, 0xba | EFI\_UGA\_DRAW\_PROTOCOL.GetMode – GetMode() with valid parameter returns EFI\_SUCCESS | 1. Call SetMode() to set 800x600x32x60 UGA mode.  2. Call GetMode() with valid parameter. The return code should be EFI\_SUCCESS |
| 5.6.3.1.3 | 0x53954b07, 0x1ee8, 0x4ab9, 0x9b, 0x5b, 0x28, 0xbe, 0xf2, 0xae, 0x65, 0x8c | EFI\_UGA\_DRAW\_PROTOCOL.GetMode – GetMode() with valid parameter returns EFI\_SUCCESS | 1. Call SetMode() to set supported UGA mode.  2. Call GetMode() with valid parameter. The return code should be EFI\_SUCCESS |
| 5.6.3.1.4 | 0xee89abe2, 0xe289, 0x4e5f, 0xbd, 0x0f, 0xee, 0x41, 0x5f, 0x9d, 0x76, 0x06 | EFI\_UGA\_DRAW\_PROTOCOL.GetMode – GetMode() with a HorizontalResolution value of NULL returns EFI\_INVALID\_PARAMETER | 1. Call GetMode() with a HorizontalResolution value of NULL. The return code should be EFI\_INVALID\_PARAMETER. |
| 5.6.3.1.5 | 0x27e72405, 0x627f, 0x4d2d, 0x8d, 0x82, 0x1c, 0xf7, 0x5a, 0x94, 0xb1, 0xe0 | EFI\_UGA\_DRAW\_PROTOCOL.GetMode – GetMode() with a VerticalResolution value of NULL returns EFI\_INVALID\_PARAMETER | 1. Call GetMode() with a VerticalResolution value of NULL. The return code should be EFI\_INVALID\_PARAMETER. |
| 5.6.3.1.6 | 0x5426aa3f, 0xcf9b, 0x49a1, 0x8b, 0x83, 0x8b, 0xd7, 0x14, 0x05, 0x68, 0x72 | EFI\_UGA\_DRAW\_PROTOCOL.GetMode – GetMode() with a RefreshRate value of NULL returns EFI\_INVALID\_PARAMETER | 1. Call GetMode() with a RefreshRate value of NULL. The return code should be EFI\_INVALID\_PARAMETER. |
| 5.6.3.1.7 | 0x36ebe5d4, 0xe938, 0x4859, 0xaa, 0x3e, 0xac, 0xe4, 0x49, 0xba, 0x5f, 0x17 | EFI\_UGA\_DRAW\_PROTOCOL.GetMode – GetMode() with a ColorDepth value of NULL returns EFI\_INVALID\_PARAMETER | 1. Call GetMode() with a ColorDepth value of NULL. The return code should be EFI\_INVALID\_PARAMETER. |

* SetMode()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.3.2.1 | 0x6a5e8496, 0x0edf, 0x4616, 0x83, 0x9f, 0xde, 0xb5, 0xf8, 0xbe, 0xc8, 0xfd | EFI\_UGA\_DRAW\_PROTOCOL.SetMode - SetMode() with supported UGA mode clears hardware frame buffer to black. | 1. Call SetMode() to set supported UGA mode.  2. Call Blt() with EfiUgaVideoToBltBuffer operation to store screen display to buffer.  3. Each pixel in buffer should be (0,0,0). |
| 5.6.3.2.2 | 0x7ff20bb2, 0xb6e7, 0x47cc, 0x86, 0xc8, 0x81, 0x7d, 0xb0, 0x73, 0x20, 0x41 | EFI\_UGA\_DRAW\_PROTOCOL.SetMode - SetMode() with resolution 800\*600 color depth 32-bit and 60 refresh rate UGA mode returns EFI\_SUCCESS. | 1. Call SetMode() to set 800x600x32x60 UGA mode. The return code must be EFI\_SUCCESS. |
| 5.6.3.2.3 | 0xa5caad17, 0x8605, 0x473a, 0xab, 0x08, 0x6b, 0x87, 0x3f, 0x81, 0x2c, 0x14 | EFI\_UGA\_DRAW\_PROTOCOL.SetMode – GetMode() returns the values set by SetMode(). | 1. Call SetMode() to set 800x600x32x60 UGA mode. The return code must be EFI\_SUCCESS.  2. Call GetMode() with valid parameter. The return values should equal to the values set by SetMode() |
| 5.6.3.2.4 | 0x7d0e59bb, 0x54a3, 0x48c8, 0x85, 0xec, 0xad, 0x89, 0xeb, 0xe6, 0x8b, 0x49 | EFI\_UGA\_DRAW\_PROTOCOL.SetMode – GetMode() returns the values set by SetMode(). | 1. Call SetMode() to set supported UGA mode. The return code must be EFI\_SUCCESS.  2. Call GetMode() with valid parameter. The return values should equal to the values set by SetMode() |
| 5.6.3.2.5 | 0x86cc4728, 0x6884, 0x4743, 0x8b, 0x3b, 0x5c, 0x95, 0x5e, 0x9a, 0x77, 0x29 | EFI\_UGA\_DRAW\_PROTOCOL.SetMode - SetMode() with valid parameters returns EFI\_SUCCESS or EFI\_UNSUPPORTED. | 1. Call SetMode() to set UGA mode. The return code must be EFI\_SUCCESS or EFI\_UNSUPPORTED. |
| 5.6.3.2.6 | 0xe1e7967e, 0xc92a, 0x42dd, 0x93, 0xce, 0xb5, 0x1d, 0x1c, 0xe0, 0x92, 0x17 | EFI\_UGA\_DRAW\_PROTOCOL.SetMode - SetMode() with supported UGA mode returns EFI\_SUCCESS. | 1. Call SetMode() to restore original UGA mode. The return code must be EFI\_SUCCESS. |

* Blt()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.3.3.1 | 0xd0bc9db6, 0xc66e, 0x46ed, 0xae, 0x61, 0x6a, 0x90, 0x28, 0x63, 0x1d, 0x34 | EFI\_UGA\_DRAW\_PROTOCOL.Blt - Blt() with EfiUgaVideoFill operation fills display rectangle with input pixel value. | 1. Call Blt() with EfiUgaVideoFill operation.  2. Call Blt() with EfiUgaVideoToBltBuffer operation to store whole video display to buffer.  3. Each pixel in the display rectangle (DestinationX, DestinationY)(DestinationX + Width, DestinationY + Height) should be equal to the input pixel BltBuffer(0,0). |
| 5.6.3.3.2 | 0xb567d336, 0xca3a, 0x474c, 0xaa, 0x84, 0xa7, 0xb4, 0xad, 0x61, 0x57, 0x58 | EFI\_UGA\_DRAW\_PROTOCOL.Blt - Blt() with EfiUgaVideoFill operation returns EFI\_SUCCESS. | 1. Call Blt() with EfiUgaVideoFill operation. The return code should be EFI\_SUCCESS. |
| 5.6.3.3.3 | 0x367d6e99, 0x6a11, 0x4d0f, 0xbf, 0x99, 0x7f, 0xbe, 0x43, 0x8b, 0x31, 0x57 | EFI\_UGA\_DRAW\_PROTOCOL.Blt - Blt() with BltEfiUgaVideoToBltBuffer operation returns EFI\_SUCCESS. | 1. Call Blt() with BltEfiUgaVideoToBltBuffer operation to store display to Buffer1. The return code should be EFI\_SUCCESS. |
| 5.6.3.3.4 | 0x85edb629, 0x147d, 0x40b0, 0x94, 0x88, 0x18, 0x02, 0x71, 0x78, 0x09, 0xcf | EFI\_UGA\_DRAW\_PROTOCOL.Blt - Blt() with BltEfiUgaVideoToBltBuffer operation returns EFI\_SUCCESS. | 1. Call Blt() with BltEfiUgaBltBufferToVideo operation to copy Buffer1 contents to video.  1. Call Blt() with BltEfiUgaVideoToBltBuffer operation to store video display in Buffer2. The return code should be EFI\_SUCCESS. |
| 5.6.3.3.5 | 0xc776eb3a, 0x6632, 0x425d, 0xb7, 0x04, 0xfa, 0xfb, 0xce, 0x1e, 0x1d, 0x0c | EFI\_UGA\_DRAW\_PROTOCOL.Blt - Blt() with BltEfiUgaBltBufferToVideo operation returns EFI\_SUCCESS. | 1. Call Blt() with BltEfiUgaBltBufferToVideo operation to copy Buffer1 contents to video display. The return code should be EFI\_SUCCESS. |
| 5.6.3.3.6 | 0x92a04254, 0x6cbe, 0x45be, 0x87, 0xc4, 0x38, 0xd4, 0x66, 0x66, 0x11, 0xe6 | EFI\_UGA\_DRAW\_PROTOCOL.Blt - Blt() with BltEfiUgaVideoToBltBuffer and BltEfiUgaBltBufferToVideo operation gets the same content of display rectangle and buffer. | 1. Call Blt() to output a blue rectangle on screen and call Blt() with BltEfiUgaVideoToBltBuffer operation to store display to Buffer1.  2. Call Blt() with BltEfiUgaBltBufferToVideo operation to copy Buffer1 to video.  3. Call Blt() with BltEfiUgaVideoToBltBuffer to store display to Buffer2.  4. Compare Buffer1 and Buffer2. Each pixel should be the same. |
| 5.6.3.3.7 | 0x9efc6f31, 0x1cb1, 0x458f, 0x9a, 0x15, 0xe3, 0x47, 0xa8, 0x36, 0x8d, 0xd8 | EFI\_UGA\_DRAW\_PROTOCOL.Blt - Blt() with EfiUgaVideoToVideo operation returns EFI\_SUCCESS. | 1. Call Blt() to output a blue rectangle on screen and call Blt() with EfiUgaVideoToVideo operation to copy source display rectangle to destination display destination. |
| 5.6.3.3.8 | 0x09777d6a, 0x14aa, 0x41eb, 0xb8, 0xbc, 0x0d, 0xcb, 0x90, 0xf6, 0x22, 0xbc | EFI\_UGA\_DRAW\_PROTOCOL.Blt - Blt() with EfiUgaVideoToVideo operation returns the same contents between source display rectangle and destination display destination. | 1. Call Blt() to output a blue rectangle on screen and call Blt() with EfiUgaVideoToVideo operation to copy source display rectangle to destination display destination.  2. Call Blt() with BltEfiUgaVideoToBltBuffer to store source display rectangle to Buffer1.  3. Call Blt() with BltEfiUgaVideoToBltBuffer to store destination display rectangle to Buffer2.  4. Compare Buffer1 and Buffer2. Each pixel should be same. |
| 5.6.3.3.9 | 0xa077b57a, 0x2d0f, 0x4d26, 0x9e, 0x41, 0x13, 0xb2, 0x6e, 0x28, 0xed, 0xe7 | EFI\_UGA\_DRAW\_PROTOCOL.Blt - Blt() with invalid BltOperation returns EFI\_INVALID\_PARAMETER. | 1. Call Blt() with invalid BltOperation. The return code should be EFI\_INVALID\_PARAMETER. |

* EFI\_SCSI\_PASS\_THRU\_PROTOCOL Test

Reference Document:

*UEFI Specification,* EFI\_SCSI\_PASS\_THRU\_PROTOCOL Section.

* PassThru()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.9.1.1.1 | 0x23512eed, 0x301c, 0x493d, 0x8a, 0x03, 0xa6, 0xd4, 0x22, 0x1b, 0xee, 0x9c | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.PassThru - Invoks PassThru() with NULL Event will verify interface correctness by returning EFI\_SUCCESS. | Call GetNextDevice() to get valid Target and Lun. Use the Target and Lun gotten before to call PassThru() with NULL Event. The return status should be EFI\_SUCCESS. |
| 5.9.1.1.2 | 0x00718d3e, 0x788a, 0x4882, 0x80, 0xf7, 0x71, 0xb4, 0xf0, 0xcf, 0x6b, 0x30 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.PassThru - Invoks PassThru() with Event will verify interface correctness by returning EFI\_SUCCESS. | Call GetNextDevice() to get valid Target and Lun. Use the Target and Lun gotten before to call PassThru() with Event. The return status should be EFI\_SUCCESS and the event should be invoked. |
| 5.9.1.1.3 | 0x4751f323, 0x0687, 0x47b6, 0xbe, 0x16, 0x57, 0x73, 0xc1, 0xa3, 0x6d, 0x28 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.PassThru - Calling PassThru() with with too long a TransferLength returns EFI\_BAD\_BUFFER\_SIZE. | Call PassThru() with the TransferLength larger than the SCSI controller can handle. It should return EFI\_BAD\_BUFFER\_SIZE and the TransferLength will be updated to the length that SCSI controller can handle. |
| 5.9.1.1.4 | 0x831dd6e6, 0x1960, 0x4c27, 0xab, 0xef, 0x2c, 0x3c, 0x0d, 0x58, 0x68, 0x7f | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.PassThru - Calling PassThru() with an invalid Target returns EFI\_INVALID\_PARAMETER. | Call PassThru() with an invalid Target. It should return EFI\_INVALID\_PARAMETER. |
| 5.9.1.1.5 | 0x8dc5b229, 0xb838, 0x4a90, 0xb3, 0x50, 0x81, 0x3c, 0x42, 0xd4, 0x85, 0x44 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.PassThru - Calling PassThru() with an invalid Lun returns EFI\_INVALID\_PARAMETER. | Call PassThru() with an invalid Lun. It should return EFI\_INVALID\_PARAMETER. |
| 5.9.1.1.6 | 0xf57be290, 0x0aa4, 0x4e8e, 0x8d, 0x09, 0xe2, 0xce, 0xbc, 0x73, 0xc0, 0x77 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.PassThru - Calling PassThru() with an invalid ScsiRequestPacket content returns EFI\_INVALID\_PARAMETER. | Call PassThru() with an invalid ScsiRequestPacket content. It should return EFI\_INVALID\_PARAMETER. |

* GetNextDevice()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.9.1.2.1 | 0x4eda0492, 0x1eb2, 0x4022, 0x87, 0x1f, 0xd3, 0x95, 0x58, 0x20, 0x1d, 0x01 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.GetnextDevice – GetnextDevice() retrieves the list of legal Target IDs and LUNs for SCSI devices on a SCSI channel. | Call GetNextDevice() with Target’s value of 0xFFFFFFFF to get the first SCSI device present on a SCSI channel. Use the Target and **Lun** which were returned to get the next SCSI device until the end. Every call of GetNextDevice() should return EFI\_SUCCESS except the last one. The last call should return EFI\_NOT\_FOUND. |
| 5.9.1.2.2 | 0x3661f513, 0xd0ea, 0x47f2, 0x8a, 0xb7, 0xaa, 0xb4, 0x6b, 0xcd, 0x93, 0xa0 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.GetnextDevice – GetnextDevice() uses former Target and Lun to get next device. | Call GetNextDevice() with Target=0xFFFFFFFF to get the first device. Then call it again to get the next device. Use the Target and Lun return from the first call to call the function. It should return EFI\_INVALID\_PARAMETER. |
| 5.9.1.2.3 | 0xd2d48206, 0xf2dd, 0x40b3, 0xaf, 0x67, 0xe9, 0xae, 0x60, 0xc7, 0x2b, 0x9f | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.GetnextDevice - Call **GetNextDevice()** with an invalid Target. | Call GetNextDevice() with an invalid Target. It should return EFI\_INVALID\_PARAMETER. |
| 5.9.1.2.4 | 0xe7e16f25, 0xca2d, 0x4de5, 0x9f, 0xf4, 0xe4, 0xcc, 0xac, 0x9d, 0xf6, 0x90 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.GetnextDevice - Call **GetNextDevice()** with an invalid **Lun**. | Call GetNextDevice() with an invalid Lun. It should return EFI\_INVALID\_PARAMETER. |

* BuildDevicePath()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.9.1.3.1 | 0x93c4def4, 0x7854, 0x42b3, 0x81, 0xbc, 0xa0, 0x4c, 0x0f, 0xd7, 0xb1, 0x93 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.BuildDevicePath - Invoks BuildDevicePath() will verify interface correctness by returning EFI\_SUCCESS. | Call GetNextDevice() to get the first device’s Target and Lun. Call BuildDevicePath() with valid parameter. Free the DevicePath. It should return EFI\_SUCCESS. |
| 5.9.1.3.2 | 0xd4c6c164, 0x0198, 0x47c6, 0xb7, 0xef, 0x01, 0x0c, 0x47, 0x42, 0xc9, 0x88 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.BuildDevicePath - Calling BuildDevicePath() with an invalid Target returns EFI\_NOT\_FOUND. | Call BuildDevicePath() with an invalid Target. It should return EFI\_NOT\_FOUND. |
| 5.9.1.3.3 | 0xec077c7f, 0x114a, 0x41b1, 0x94, 0x83, 0x5b, 0x38, 0x10, 0xdb, 0xc4, 0x00 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.BuildDevicePath - Calling BuildDevicePath() with an invalid Lun returns EFI\_NOT\_FOUND. | Call BuildDevicePath() with an invalid Lun. It should return EFI\_NOT\_FOUND. |
| 5.9.1.3.4 | 0x8a1ce910, 0x8a20, 0x4a72, 0xb7, 0x05, 0xb8, 0x09, 0x70, 0xc7, 0xdf, 0xd3 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.BuildDevicePath - Calling BuildDevicePath() with NULL DevicePath returns EFI\_INVALID\_PARAMETER. | Call BuildDevicePath() with a. NULL DevicePath. It should return EFI\_INVALID\_PARAMETER. |

* GetTargetLun()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.9.1.4.1 | 0x8d06f9c5, 0xd470, 0x4b31, 0xbe, 0xb9, 0x73, 0x3e, 0x5d, 0x8f, 0xf4, 0xcb | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.GetTargetLun - Invoks GetTargetLun()will verify interface correctness by returning EFI\_SUCCESS. | Call GetNextDevice() and GetTargetLun() to get the valid DevicePath. Use this DevicePath to call GetTargetLun().The return code should be EFI\_SUCCESS. |
| 5.9.1.4.2 | 0x462c4098, 0xfd65, 0x4005, 0x8e, 0xdb, 0x7b, 0xb5, 0x95, 0x65, 0xc5, 0x11 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.GetTargetLun - Invoks GetTargetLun() with NULL DevicePath returns EFI\_INVALID\_PARAMETER. | Call GetTargetLun() with NULL DevicePath. It should return EFI\_INVALID\_PARAMETER. |
| 5.9.1.4.3 | 0x884c336a, 0xeffd, 0x45b3, 0xb5, 0xcb, 0xc5, 0x50, 0x2a, 0xfa, 0xcf, 0x3f | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.GetTargetLun - Invoks GetTargetLun() with NULL Target returns EFI\_INVALID\_PARAMETER. | Call GetTargetLun() with NULL Target. It should return EFI\_INVALID\_PARAMETER. |
| 5.9.1.4.4 | 0x842b366f, 0x035e, 0x46a7, 0x8f, 0x07, 0x45, 0xd8, 0xd1, 0xe1, 0xe1, 0x72 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.GetTargetLun - Invoks GetTargetLun() with NULL Lun returns EFI\_INVALID\_PARAMETER. | Call GetTargetLun() with NULL Lun. It should return EFI\_INVALID\_PARAMETER. |
| 5.9.1.4.5 | 0xf29750b2, 0xd353, 0x4baa, 0x8a, 0x44, 0x29, 0xc2, 0x4e, 0xe8, 0x49, 0x43 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.GetTargetLun - Calling GetTargetLun() with unsupported DevicePath returns EFI\_UNSUPPORTED. | Call GetTargetLun() with unsupported DevicePath. It should return EFI\_UNSUPPORTED. |

* ResetChannel()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.9.1.5.1 | 0x8af96e89, 0x2209, 0x47d9, 0x9b, 0x84, 0xa1, 0xf6, 0xf2, 0xd1, 0x8a, 0x6b | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.ResetChannel - Invoks ResetChannel() will verify interface correctness via return code of EFI\_SUCCESS or EFI\_UNSUPPORTED. | Call ResetChannel().The return code should be EFI\_SUCCESS or EFI\_UNSUPPORTED. |

* ResetTarget()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.9.1.6.1 | 0xbac42d29, 0x75cc, 0x4b9b, 0xa3, 0x16, 0xdf, 0x11, 0xca, 0x7c, 0xf1, 0xe4 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.ResetTarget - Invoks ResetTarget() will verify interface correctness via return code of EFI\_SUCCESS or EFI\_UNSUPPORTED. | Call GetNextDevice() to get valid Target and Lun. Use the Target and Lun gotten before to call ResetTarget().The return code should be EFI\_SUCCESS or EFI\_UNSUPPORTED. |
| 5.9.1.6.2 | 0x04296f40, 0xe48b, 0x4b5c, 0xb2, 0xcf, 0x49, 0x25, 0xf0, 0x98, 0x5d, 0x82 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.ResetTarget - Calling ResetTarget() with an invalid Target returns EFI\_INVALID\_PARAMETER. | Call GetResetTarget() with an invalid Target. It should return EFI\_INVALID\_PARAMETER. |
| 5.9.1.6.3 | 0xc75f3592, 0xee1a, 0x43a3, 0xaa, 0x9b, 0x08, 0x16, 0x9e, 0xca, 0xa6, 0x93 | EFI\_SCSI\_PASS\_THRU\_PROTOCOL.ResetTarget - Calling ResetTarget() with an invalid Lun returns EFI\_INVALID\_PARAMETER. | Call GetResetTarget() with an invalid Lun. It should return EFI\_INVALID\_PARAMETER. |