# Protocols USB Support Test

## EFI\_USB2\_HC\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_USB2\_HC\_PROTOCOL Section.

Most of functionalities rely on the real USB devices. They are not covered in below checkpoints.

### GetCapability()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.1.1 | 0xbe0fffbd, 0xc5cb, 0x4ab7, 0xa0, 0x8a, 0x79, 0xd1, 0x02, 0xb3, 0x5f, 0xf8 | EFI\_USB2\_HC\_PROTOCOL. GetCapability - GetCapability() returns EFI\_INVALID\_PARAMETER with aMaxSpeed value of NULL. | 1. Call GetCapability() with a MaxSpeedvalue of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.1.2 | 0x6dd53bd5, 0x463b, 0x46a7, 0xb0, 0x98, 0x06, 0xa6, 0xf6, 0xa5, 0x62, 0xdd | EFI\_USB2\_HC\_PROTOCOL. GetCapability - GetCapability () returns EFI\_INVALID\_PARAMETER with a PortNumber value of NULL. | 1. Call GetCapability () with a PortNumber value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.1.3 | 0x0ffa5751, 0x96dd, 0x4a70, 0xa1, 0x01, 0x63, 0x66, 0x7b, 0x15, 0xcc, 0xf5 | EFI\_USB2\_HC\_PROTOCOL. GetCapability - GetCapability () returns EFI\_INVALID\_PARAMETER with an Is64BitCapablevalue of NULL. | 1. Call GetCapability () with an Is64BitCapable value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |

### Reset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.2.1 | 0xf8dd84cb, 0x72a2, 0x4cab, 0xac, 0x2e, 0x11, 0x6f, 0x3c, 0x0d, 0x5d, 0xb5 | EFI\_USB2\_HC\_PROTOCOL.Reset - Reset**()** returns EFI\_SUCCESS with Attributes values of EFI\_USB\_HC\_RESET\_GLOBAL. | 1. Call Reset() with Attributes values of EFI\_USB\_HC\_RESET\_GLOBAL. The return status should be EFI\_SUCCESS.  2. Call GetState() to get the state of the USB host controller. The controller should be in halt state. |
| 5.21.1.2.2 | 0x3bdb0674, 0x621b, 0x4319, 0xb2, 0x4f, 0xc6, 0x1a, 0xd4, 0x09, 0x73, 0xd0 | EFI\_USB2\_HC\_PROTOCOL.Reset - Reset**()** returns EFI\_SUCCESS with Attributes values of EFI\_USB\_HC\_RESET\_HOST\_CONTROLLER. | 1. Call Reset() with Attributes values of EFI\_USB\_HC\_RESET\_HOST\_CONTROLLER. The return status should be EFI\_SUCCESS.  2. Call GetState() to get the state of the USB host controller. The controller should be in halt state. |
| 5.21.1.2.3 | 0xd243c0fd, 0x7654, 0x4400, 0xb3, 0x4a, 0xe3, 0x09, 0x8f, 0x9e, 0x5e, 0xd4 | EFI\_USB2\_HC\_PROTOCOL.Reset - Reset**()** returns EFI\_SUCCESS with Attributes values of EFI\_USB\_HC\_RESET\_GLOBAL | EFI\_USB\_HC\_RESET\_HOST\_CONTROLLER. | 1. Call Reset() with Attributes values of EFI\_USB\_HC\_RESET\_GLOBAL | EFI\_USB\_HC\_RESET\_HOST\_CONTROLLER. The return status should be EFI\_SUCCESS.  2. Call GetState() to get the state of the USB host controller. The controller should be in halt state. |
| 5.21.1.2.4 | 0xa4f18be1, 0x15f2, 0x424f, 0xa6, 0xdb, 0x58, 0x6e, 0x0d, 0x54, 0x80, 0x25 | EFI\_USB2\_HC\_PROTOCOL.Reset - Reset**()** returns EFI\_SUCCESS with Attributes values of EFI\_USB\_HC\_RESET\_GLOBAL\_DEBUG. | 1. Call Reset() with Attributes values of EFI\_USB\_HC\_RESET\_GLOBAL\_DEBUG. The return status should be EFI\_SUCCESS.  2. Call GetState() to get the state of the USB host controller. The controller should be in halt state. |
| 5.21.1.2.5 | 0xe2df74c7, 0x7aea, 0x488c, 0xb9, 0xa2, 0x71, 0x94, 0xb2, 0x5f, 0xf3, 0x8b | EFI\_USB2\_HC\_PROTOCOL.Reset - Reset**()** returns EFI\_SUCCESS with Attributes values of EFI\_USB\_HC\_RESET\_HOST\_CONTROLLER\_DEBUG. | 1. Call Reset() with an Attributes value of EFI\_USB\_HC\_RESET\_HOST\_CONTROLLER\_DEBUG. The return status should be EFI\_SUCCESS.  2. Call GetState() to get the state of the USB host controller. The controller should be in halt state. |
| 5.21.1.2.6 | 0xda7ef15c, 0x01a4, 0x4004, 0x8c, 0x7a, 0x33, 0xc7, 0x89, 0x47, 0xd9, 0xfc | EFI\_USB2\_HC\_PROTOCOL.Reset - Reset**()** returns EFI\_SUCCESS with an Attributes value of EFI\_USB\_HC\_RESET\_GLOBAL\_DEBUG | EFI\_USB\_HC\_RESET\_HOST\_CONTROLLER\_DEBUG. | 1. Call Reset() with an Attributes value of EFI\_USB\_HC\_RESET\_GLOBAL\_DEBUG | EFI\_USB\_HC\_RESET\_HOST\_CONTROLLER\_DEBUG. The return status should be EFI\_SUCCESS.  2. Call GetState() to get the state of this USB host controller. This controller should be in halt state. |
| 5.21.1.2.7 | 0xd2e6a8f0, 0x6c97, 0x4134, 0x81, 0x2e, 0x25, 0xf1, 0x75, 0x18, 0x6a, 0xe4 | EFI\_USB2\_HC\_PROTOCOL.Reset - Reset**()** returns EFI\_INVALID\_PARAMETER with an invalid Attributes. | 1. Call Reset() with an invalid Attributes value of 0. The return status should be EFI\_INVALID\_PARAMETER. |

### GetState()

|  |  |  |  |
| --- | --- | --- | --- |
| 5.21.1.3.1 | 0x19be62be, 0xf20c, 0x4fa2, 0x89, 0xcc, 0x3a, 0x89, 0x39, 0x48, 0x4d, 0x86 | EFI\_USB2\_HC\_PROTOCOL.GetState - GetState() returns EFI\_SUCCESS while the host controller is in halt state. | 1. Call SetState() with a State value of EfiUsbHcStateHalt. The return status should be EFI\_SUCCESS.  2. Call GetState() to get the state of this USB host controller. This controller should be in halt state. |
| 5.21.1.3.2 | 0xc2b1cb6a, 0x66b4, 0x4c6d, 0xb9, 0x0a, 0xc9, 0x5d, 0x27, 0xd6, 0xa5, 0xd1 | EFI\_USB2\_HC\_PROTOCOL.GetState - GetState() returns EFI\_SUCCESS while the host controller is in an operational state. | 1. Call SetState() with a State value of EfiUsbHcStateOperational. The return status should be EFI\_SUCCESS.  2. Call GetState() to get the state of this USB host controller. This controller should be in an Operational state. |
| 5.21.1.3.3 | 0x95e913a0, 0x5ca9, 0x4edb, 0x92, 0x4f, 0xaa, 0x2f, 0x18, 0x9b, 0x57, 0x6a | EFI\_USB2\_HC\_PROTOCOL.GetState - GetState() returns EFI\_SUCCESS while the host controller is in suspend state. | 1. Call SetState() with a State value of EfiUsbHcStateSuspend. The return status should be EFI\_SUCCESS.  2. Call GetState() to get the state of this USB host controller. This controller should be in Suspend state. |
| 5.21.1.3.4 | 0xbc1b8f2e, 0xf1aa, 0x446f, 0x81, 0x78, 0x6e, 0x4e, 0xd5, 0x53, 0x02, 0x08 | EFI\_USB2\_HC\_PROTOCOL.GetState - GetState() returns EFI\_INVALID\_PARAMETER with a **State** value of NULL. | 1. Call GetState() with a State value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |

### SetState()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.4.1 | 0x5d2282fe, 0xc37c, 0x4901, 0xbb, 0xf7, 0xf1, 0xb6, 0xf0, 0xae, 0x82, 0x91 | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_SUCCESS when changing the state from halt to halt. | 1. Call SetState() with a State value of EfiUsbHcStateHalt. The return status should be EFI\_SUCCESS.  2. Call SetState() with a State value of EfiUsbHcStateHalt again. The return status should be EFI\_SUCCESS.  3. Call GetState() to get the state of this USB host controller. This controller should be in halt state. |
| 5.21.1.4.2 | 0x6f6e6713, 0x07dc, 0x4413, 0x85, 0x05, 0xee, 0x69, 0x9e, 0x32, 0x69, 0x27 | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_SUCCESS when changing the state from halt state to operational state. | 1. Call SetState() with a State value of EfiUsbHcStateHalt. The return status should be EFI\_SUCCESS.  2. Call SetState() with a State value of EfiUsbHcStateOperational again. The return status should be EFI\_SUCCESS.  3. Call GetState() to get the state of this USB host controller. This controller should be in an Operational state. |
| 5.21.1.4.3 | 0x49ca37bc, 0x208d, 0x4feb, 0xa6, 0xd9, 0x68, 0xa3, 0x69, 0xca, 0xb3, 0xf1 | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_SUCCESS when changing the state from halt state to suspend state. | 1. Call SetState() with a State value of EfiUsbHcStateHalt. The return status should be EFI\_SUCCESS.  2. Call SetState() with a State value of EfiUsbHcStateSuspend again. The return status should be EFI\_SUCCESS.  3. Call GetState() to get the state of this USB host controller. This controller should be in Suspend state. |
| 5.21.1.4.4 | 0xa4663706, 0xd0c0, 0x45d7, 0x9a, 0x9d, 0x5e, 0x0e, 0xf8, 0xba, 0x2c, 0x26 | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_SUCCESS when changing the state from operational state to operational state. | 1. Call SetState() with a State value of EfiUsbHcStateOperational. The return status should be EFI\_SUCCESS.  2. Call SetState() with a State value of EfiUsbHcStateOperational again. The return status should be EFI\_SUCCESS.  3. Call GetState() to get the state of this USB host controller. This controller should be in an Operational state. |
| 5.21.1.4.5 | 0xa9b73b45, 0xb3ca, 0x4579, 0x87, 0x38, 0xb3, 0xcc, 0xc4, 0x50, 0x09, 0x97 | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_SUCCESS when changing the state from operational state to halt state. | 1. Call SetState() with a State value of EfiUsbHcStateOperational. The return status should be EFI\_SUCCESS.  2. Call SetState() with a State value of EfiUsbHcStateHalt again. The return status should be EFI\_SUCCESS.  3. Call GetState() to get the state of this USB host controller. This controller should be in halt state. |
| 5.21.1.4.6 | 0x54936ebc, 0x9732, 0x4d9f, 0x83, 0x5c, 0x95, 0x77, 0xf5, 0xdb, 0x0e, 0xb1 | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_SUCCESS when changing the state from operational state to suspend state. | 1. Call SetState() with a State value of EfiUsbHcStateOperational. The return status should be EFI\_SUCCESS.  2. Call SetState() with a State value of EfiUsbHcStateSuspend again. The return status should be EFI\_SUCCESS.  3. Call GetState() to get the state of this USB host controller. This controller should be in Suspend state. |
| 5.21.1.4.7 | 0x9da57b17, 0x7841, 0x423a, 0xb1, 0xf8, 0x6d, 0x61, 0xf0, 0xd3, 0x17, 0xf0 | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_SUCCESS when changing the state from suspend state to suspend state. | 1. Call SetState() with a State value of EfiUsbHcStateSuspend. The return status should be EFI\_SUCCESS.  2. Call SetState() with a State value of EfiUsbHcStateSuspend again. The return status should be EFI\_SUCCESS.  3. Call GetState() to get the state of this USB host controller. This controller should be in Suspend state. |
| 5.21.1.4.8 | 0x5b4bf27e, 0xad64, 0x41a4, 0xa9, 0x8b, 0xd2, 0xb0, 0x7d, 0x32, 0xbb, 0xa3 | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_SUCCESS when changing the state from suspend state to halt state. | 1. Call SetState() with a State value of EfiUsbHcStateSuspend. The return status should be EFI\_SUCCESS.  2. Call SetState() with a State value of EfiUsbHcStateHalt again. The return status should be EFI\_SUCCESS.  3. Call GetState() to get the state of this USB host controller. This controller should be in halt state. |
| 5.21.1.4.9 | 0xc12e9ca0, 0x0e9c, 0x4204, 0xaa, 0xc3, 0x6d, 0x12, 0x33, 0x1b, 0x28, 0x9b | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_SUCCESS when changing the state from suspend state to operational state. | 1. Call SetState() with a State value of EfiUsbHcStateSuspend. The return status should be EFI\_SUCCESS.  2. Call SetState() with a State value of EfiUsbHcStateOperational again. The return status should be EFI\_SUCCESS.  3. Call GetState() to get the state of this USB host controller. This controller should be in an Operational state. |
| 5.21.1.4.10 | 0x5168c4ef, 0x91f4, 0x48c5, 0x88, 0x1f, 0xf8, 0x01, 0x80, 0xd2, 0x98, 0x07 | EFI\_USB2\_HC\_PROTOCOL.SetState - SetState() returns EFI\_INVALID\_PARAMETER with an invalid **State**. | 1. Call SetState() with an invalid State value of -1. The return status should be EFI\_INVALID\_PARAMETER.  2. Call SetState() with an invalid State value of EfiUsbHcStateMaximum. The return status should be EFI\_INVALID\_PARAMETER. |

### ControlTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.5.1 | 0x36308487, 0x3a2c, 0x48fa, 0x91, 0xed, 0xec, 0xc3, 0x59, 0xd0, 0x78, 0x46 | EFI\_USB2\_HC\_PROTOCOL. ControlTransfer - ControlTransfer () returns EFI\_INVALID\_PARAMETER with an invalid TransferDirection. | 1. Call ControlTransfer() with an invalid TransferDirection value  (-**1**). The return status should be EFI\_INVALID\_PARAMETER.  2. Call ControlTransfer() with an invalid TransferDirection value (**0x7FFFFFFF***)*. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.5.2 | 0x26532efd, 0x62ab, 0x4d60, 0x9c, 0xd8, 0x14, 0xb7, 0x9d, 0x48, 0x8e, 0xa1 | EFI\_USB2\_HC\_PROTOCOL. ControlTransfer - ControlTransfer () returns EFI\_INVALID\_PARAMETER with a invalid Dataand DataLength values. | 1. Call ControlTransfer() with an invalid **Data** (value of*)* and TransferDirection is either EfiUsbDataIn or EfiUsbDataOut. The return status should be EFI\_INVALID\_PARAMETER.  2. Call ControlTransfer() with an invalid DataLength (value of **0***)* and TransferDirection is either EfiUsbDataIn or EfiUsbDataOut. The return status should be EFI\_INVALID\_PARAMETER.  3. Call ControlTransfer() with an invalid Data **(**not value of NULL*)* and TransferDirection value of EfiUsbNoData.The return status should be EFI\_INVALID\_PARAMETER.  4. Call ControlTransfer() with an invalid DataLength**(** value of 1*)*and TransferDirection value of EfiUsbNoData.The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.5.3 | 0x28f002fd, 0x3797, 0x46cb, 0xaf, 0x66, 0xd5, 0xb4, 0x27, 0x23, 0x1b, 0x7a | EFI\_USB2\_HC\_PROTOCOL. ControlTransfer - ControlTransfer () returns EFI\_INVALID\_PARAMETER with an Request value of NULL. | 1. Call ControlTransfer() with an invalid Request value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.5.4 | 0xddf99154, 0x12ea, 0x4c99, 0x9a, 0x49, 0x6a, 0x1c, 0x51, 0xc2, 0x7a, 0x77 | EFI\_USB2\_HC\_PROTOCOL. ControlTransfer - ControlTransfer () returns EFI\_INVALID\_PARAMETER with an invalid MaximumPacketLength. | 1. Call ControlTransfer() with an invalid MaximumPacketLength ( value is not 8) when DeviceSpeed is EFI\_USB\_SPEED\_LOW. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.5.5 | 0xc258056b, 0x13ae, 0x4839, 0xbb, 0xda, 0xa0, 0x1f, 0x5c, 0x14, 0x0a, 0x51 | EFI\_USB2\_HC\_PROTOCOL. ControlTransfer - ControlTransfer () returns EFI\_INVALID\_PARAMETER with an invalid MaximumPacketLength. | 1. Call ControlTransfer() with an invalid MaximumPacketLength (value of 128 not 8/16/32/64) when DeviceSpeed is EFI\_USB\_SPEED\_FULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.5.6 | 0x5f6973f9, 0x9d75, 0x4e26, 0x8a, 0x30, 0xb5, 0xc2, 0x0e, 0x47, 0xf0, 0xb3 | EFI\_USB2\_HC\_PROTOCOL. ControlTransfer - ControlTransfer () returns EFI\_INVALID\_PARAMETER with an invalid MaximumPacketLength. | 1. Call ControlTransfer() with an invalid MaximumPacketLength (value of 128 not 8/16/32/64) when DeviceSpeed is EFI\_USB\_SPEED\_HIGH. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.5.7 | 0x66a39c82, 0xfb44, 0x4057, 0xbb, 0xd7, 0x4b, 0x24, 0x30, 0xff, 0x19, 0xa9 | EFI\_USB2\_HC\_PROTOCOL. ControlTransfer - ControlTransfer () returns EFI\_INVALID\_PARAMETER with an invalid MaximumPacketLength. | 1. Call ControlTransfer() with an invalid MaximumPacketLength*(*value of 256 not 512*)* when DeviceSpeed is EFI\_USB\_SPEED\_SUPER. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.5.8 | 0xf63896ea, 0x5143, 0x4b7a, 0x93, 0x51, 0x63, 0xb5, 0xb5, 0x95, 0x81, 0x5c | EFI\_USB2\_HC\_PROTOCOL. ControlTransfer - ControlTransfer () returns EFI\_INVALID\_PARAMETER with a TransferResult value of NULL. | 1. Call ControlTransfer() with an invalid TransferResult (value of NULL). The return status should be EFI\_INVALID\_PARAMETER. |

### BulkTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.6.1 | 0x0498c13e, 0xc21b, 0x4c4e, 0x95, 0xd2, 0x11, 0x9a, 0x10, 0x07, 0x51, 0x02 | EFI\_USB2\_HC\_PROTOCOL. BulkTransfer - BulkTransfer () returns EFI\_INVALID\_PARAMETER with a Data value of NULL. | 1. Call BulkTransfer() with an invalid Data value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.6.2 | 0x2a1df585, 0xf82a, 0x42ab, 0x97, 0x4f, 0xfe, 0xfb, 0xf7, 0x89, 0xe6, 0xf5 | EFI\_USB2\_HC\_PROTOCOL. BulkTransfer - BulkTransfer () returns EFI\_INVALID\_PARAMETER with a DataLength value of 0. | 1. Call BulkTransfer() with an invalid DataLength value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.6.3 | 0x26ad2292, 0x449b, 0x4545, 0x80, 0xaa, 0x13, 0x39, 0x13, 0x15, 0x04, 0xf6 | EFI\_USB2\_HC\_PROTOCOL. BulkTransfer - BulkTransfer () returns EFI\_INVALID\_PARAMETER with a DeviceSpeed value of EFI\_USB\_SPEED\_LOW. | 1. Call BulkTransfer() with an invalid DeviceSpeed value of EFI\_USB\_SPEED\_LOW. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.6.4 | 0x1d89742e, 0xd026, 0x47d7, 0xa4, 0xcb, 0xe0, 0xb6, 0xd9, 0xc3, 0xd9, 0x54 | EFI\_USB2\_HC\_PROTOCOL. BulkTransfer - BulkTransfer () returns EFI\_INVALID\_PARAMETER with an invalid MaximumPacketLength. | 1. Call BulkTransfer() with an invalid MaximumPacketLength (value of 65) when DeviceSpeed is EFI\_USB\_SPEED\_FULL. The return status should be EFI\_INVALID\_PARAMETER.  2. Call BulkTransfer() with an invalid MaximumPacketLength (value of 513) when DeviceSpeed is EFI\_USB\_SPEED\_HIGH. The return status should be EFI\_INVALID\_PARAMETER.  3. Call BulkTransfer() with an  invalid MaximumPacketLength (value of 1025) when DeviceSpeed is EFI\_USB\_SPEED\_SUPER. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.6.5 | 0xbc90875e, 0x0a8b, 0x4e3c, 0xbb, 0xf2, 0x5a, 0x43, 0x40, 0x3a, 0x6b, 0x05 | EFI\_USB2\_HC\_PROTOCOL. BulkTransfer - BulkTransfer () returns EFI\_INVALID\_PARAMETER with a DataToggle value other than 0 and 1. | 1. Call BulkTransfer() with an invalid DataToggle(value of 2*)*. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.6.6 | 0x0dfea5a1, 0xf82a, 0x41a5, 0xbf, 0x67, 0xea, 0x89, 0xed, 0x74, 0x61, 0x21 | EFI\_USB2\_HC\_PROTOCOL. BulkTransfer - BulkTransfer () returns EFI\_INVALID\_PARAMETER with a TransferResult value of NULL. | 1. Call BulkTransfer() with an invalid TransferResult ( value of NULL). The return status should be EFI\_INVALID\_PARAMETER. |

### AsyncInterruptTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.7.1 | 0xec3427c4, 0xe4df, 0x4646, 0x8b, 0x63, 0xdc, 0x0b, 0x7d, 0xc0, 0x0d, 0xdd | EFI\_USB2\_HC\_PROTOCOL. AsyncInterruptTransfer - AsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a EndPointAddress value other than EfiUsbDataIn. | 1. Call AsyncInterruptTransfer () with an *E*ndPointAddress value other than EfiUsbDataIn. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.7.2 | 0xc0cddbce, 0x4853, 0x4d71, 0xad, 0xe1, 0x59, 0x94, 0x90, 0x7c, 0x31, 0xcc | EFI\_USB2\_HC\_PROTOCOL. AsyncInterruptTransfer - AsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a new, invalid transfer. | 1. Call AsyncInterruptTransfer () with the IsNewTransfer value of TRUE and DataLength value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.7.3 | 0xaf26077c, 0x75e5, 0x4fbc, 0xad, 0x5e, 0x99, 0x3b, 0xce, 0x66, 0xb5, 0xc5 | EFI\_USB2\_HC\_PROTOCOL. AsyncInterruptTransfer - AsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a new, invalid transfer. | 1. Call AsyncInterruptTransfer () with the IsNewTransfer value of TRUE and a DataToggle value other than 0 and 1. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.7.4 | 0xccd35e94, 0x51db, 0x4118, 0xa8, 0xd4, 0x40, 0xbd, 0x2e, 0xee, 0x77, 0xd9 | EFI\_USB2\_HC\_PROTOCOL. AsyncInterruptTransfer - AsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with new, invalid transfer. | 1. Call AsyncInterruptTransfer () with the IsNewTransfer value of TRUE and PollingInterval value of 0. The return status should be EFI\_INVALID\_PARAMETER.  2. Call AsyncInterruptTransfer () with the IsNewTransfer value of TRUE and PollingInterval value of 256. The return status should be EFI\_INVALID\_PARAMETER. |

### 

### SyncInterruptTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.8.1 | 0x509cb496, 0x1d63, 0x4faf, 0x8d, 0xdf, 0x00, 0xbc, 0x58, 0x05, 0x0d, 0xe6 | EFI\_USB2\_HC\_PROTOCOL. SyncInterruptTransfer - SyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with the EndPointAddress set other than EfiUsbDataIn. | 1. Call SyncInterruptTransfer () with the EndPointAddress set other than EfiUsbDataIn. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.8.2 | 0x3a0ad565, 0xb82c, 0x450f, 0xbc, 0xe6, 0x88, 0xb3, 0xd1, 0x6a, 0xde, 0x35 | EFI\_USB2\_HC\_PROTOCOL. SyncInterruptTransfer - SyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a Data value of NULL. | 1. Call SyncInterruptTransfer () with a Data value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.8.3 | 0xc139127a, 0x3797, 0x482f, 0xb3, 0x5c, 0xaa, 0xf7, 0x99, 0xbd, 0xf6, 0xc6 | EFI\_USB2\_HC\_PROTOCOL. SyncInterruptTransfer - SyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a DataLength value of 0. | 1. Call SyncInterruptTransfer () with a DataLength value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.8.4 | 0x14cb206c, 0x422b, 0x47ee, 0x8c, 0x4b, 0xf3, 0x16, 0xfe, 0x33, 0xda, 0xfb | EFI\_USB2\_HC\_PROTOCOL. SyncInterruptTransfer - SyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid MaximumPacketLength. | 1. Call SyncInterruptTransfer () with a MaximumPacketLength value of 9 and DeviceSpeed value of EFI\_USB\_SPEED\_LOW. The return status should be EFI\_INVALID\_PARAMETER.  2. Call SyncInterruptTransfer () with a MaximumPacketLength value of 65 and DeviceSpeed value of EFI\_USB\_SPEED\_FULL. The return status should be EFI\_INVALID\_PARAMETER.  3. Call SyncInterruptTransfer () with a MaximumPacketLength value of 3073 and DeviceSpeed value of EFI\_USB\_SPEED\_HIGH. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.8.5 | 0xf4353439, 0x47e4, 0x4df3, 0x85, 0xe9, 0x9e, 0xfe, 0x72, 0x3a, 0x1e, 0x4b | EFI\_USB2\_HC\_PROTOCOL. SyncInterruptTransfer - SyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with DataToggle pointing to a value other than 0 and 1. | 1. Call SyncInterruptTransfer () with DataToggle points to a value other than 0 and 1. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.8.6 | 0x81dfdb23, 0x681e, 0x4df7, 0xa7, 0x73, 0x6d, 0x41, 0x58, 0xdb, 0x88, 0x3e | EFI\_USB2\_HC\_PROTOCOL. SyncInterruptTransfer - SyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a TransferResult value of NULL. | 1. Call SyncInterruptTransfer () with a TransferResult value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |

### IsochronousTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.9.1 | 0x74e2dcbf, 0xae9f, 0x4499, 0x82, 0x74, 0xcb, 0xbe, 0x86, 0x59, 0x5d, 0xb7 | EFI\_USB2\_HC\_PROTOCOL. IsochronousTransfer - IsochronousTransfer () returns EFI\_INVALID\_PARAMETER with a Datavalue of NULL. | 1. Call IsochronousTransfer () with a Data value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.9.2 | 0xd93babd4, 0xd7de, 0x4e87, 0x9b, 0x5c, 0x68, 0xd2, 0xa6, 0x77, 0x33, 0xc4 | EFI\_USB2\_HC\_PROTOCOL. IsochronousTransfer - IsochronousTransfer () returns EFI\_INVALID\_PARAMETER with a DataLengthvalue of 0. | 1. Call IsochronousTransfer () with a DataLength value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.9.3 | 0x9b220909, 0x662c, 0x4b5e, 0x9e, 0x42, 0xdc, 0x66, 0x4c, 0xdb, 0xb1, 0x5f | EFI\_USB2\_HC\_PROTOCOL. IsochronousTransfer - IsochronousTransfer () returns EFI\_INVALID\_PARAMETER with a MaxiumPacketLenth set larger than 1023. | 1. Call IsochronousTransfer () with a MaxiumPacketLenth value of 1024. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.9.4 | 0x68898a17, 0x5ae9, 0x456a, 0xb1, 0xe0, 0xa3, 0xc0, 0x42, 0xeb, 0x50, 0x8d | EFI\_USB2\_HC\_PROTOCOL. IsochronousTransfer - IsochronousTransfer () returns EFI\_INVALID\_PARAMETER with a TransferResultvalue of NULL. | 1. Call IsochronousTransfer () with a TransferResultvalue of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.9.5 | 0xfa4f5868, 0xf004, 0x4cbe, 0x88, 0x97, 0xfd, 0x6, 0xb2, 0x72, 0x76, 0x71 | EFI\_USB2\_HC\_PROCOTOL.IsochronousTransferIsochronousTransfer () returnsEFI\_INVALID\_PARAMETER when *DeviceSpeed* is not one of the supported values. | **IsochronousTransfer()** returns **EFI\_INVALID\_PARAMETER** when *DeviceSpeed* is not one of the supported values. |

### AsyncIsochronousTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.10.1 | 0x55a7ea0c, 0x9ffc, 0x47dc, 0xb7, 0x5e, 0x5c, 0xfa, 0x8c, 0xed, 0xe1, 0x53 | EFI\_USB2\_HC\_PROTOCOL. AsyncIsochronousTransfer - AsyncIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with a Datavalue of NULL. | 1. Call AsyncIsochronousTransfer () with a Data value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.10.2 | 0xfa310dd6, 0x4b8a, 0x4799, 0xa5, 0xdc, 0x80, 0xe7, 0xbb, 0xe0, 0x4e, 0xac | EFI\_USB2\_HC\_PROTOCOL. AsyncIsochronousTransfer - AsyncIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with a DataLengthvalue of 0. | 1. Call AsyncIsochronousTransfer () with a DataLength value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.10.3 | 0x4083742a, 0x6c43, 0x49b4, 0x8d, 0xe1, 0x7a, 0xf8, 0x0c, 0x8b, 0x02, 0x33 | EFI\_USB2\_HC\_PROTOCOL. AsyncIsochronousTransfer - AsyncIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with a MaxiumPacketLenth value of larger than 1023. | 1. Call AsyncIsochronousTransfer () with a MaxiumPacketLenth value of 1024. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.10.4 | 0x474590c4, 0x8410, 0x4871, 0x93, 0xb4, 0x2b, 0xe, 0x9f, 0xb5, 0xe8, 0x30 | USB2\_HC\_PROCOTOL.AsyncIsochronousTransfer –AsyncIsochronousTransfer () returns EFI\_INVALID\_PARAMETERwhen *DeviceSpeed* is not one of the supported values. | **AsyncIsochronousTransfer()** returns **EFI\_INVALID\_PARAMETER** when *DeviceSpeed* is not one of the supported values. |

### GetRootHubPortStatus()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.11.1 | 0x089705c5, 0xf134, 0x42b4, 0xbd, 0xeb, 0x7a, 0x74, 0xc7, 0x93, 0xa0, 0xf5 | EFI\_USB2\_HC\_PROTOCOL. GetRootHubPortStatus - GetRootHubPortStatus () returns EFI\_INVALID\_PARAMETER with an invalid PortNumber. | 1. Call GetCapability() to get the number of ports. The return status should be EFI\_SUCCESS.  2. Call GetRootHubPortStatus() with PortNumber greater than the number of ports. The return status should be EFI\_INVALID\_PARAMETER. |

### SetRootHubPortFeature()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.12.1 | 0xf74da277, 0x4ac2, 0x422c, 0x90, 0xda, 0xb4, 0x9f, 0xc7, 0x4f, 0x2a, 0x65 | EFI\_USB2\_HC\_PROTOCOL. SetRootHubPortFeature - SetRootHubPortFeature () returns EFI\_INVALID\_PARAMETER with an invalid PortNumber. | 1. Call GetRootHubPortNumber () to get the number of ports. The return status should be EFI\_SUCCESS.  2. Call SetRootHubPortFeature() with a PortNumber greater than the number of ports. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.12.2 | 0xd7071255, 0x61db, 0x446a, 0xad, 0x65, 0x01, 0xb4, 0x54, 0x72, 0x1f, 0x80 | EFI\_USB2\_HC\_PROTOCOL. SetRootHubPortFeature - SetRootHubPortFeature () returns EFI\_INVALID\_PARAMETER with an invalid PortFeature. | 1. Call GetRootHubPortNumber () to get the number of ports. The return status should be EFI\_SUCCESS.  2. Call SetRootHubPortFeature() with a PortFeature not value of EfiUsbPortEnable, EfiUsbPortSuspend, EfiUsbPortReset nor EfiUsbPortPower. The return status should be EFI\_INVALID\_PARAMETER. |

### ClearRootHubPortFeature()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.1.13.1 | 0x88cda060, 0xbe70, 0x4c49, 0x95, 0xac, 0xae, 0xa0, 0x37, 0xfa, 0x7f, 0x51 | EFI\_USB2\_HC\_PROTOCOL. ClearRootHubPortFeature - ClearRootHubPortFeature () returns EFI\_INVALID\_PARAMETER with an invalid PortNumber. | 1. Call GetRootHubPortNumber () to get the number of ports. The return status should be EFI\_SUCCESS.  2. Call ClearRootHubPortFeature() with a PortNumber greater than the number of ports. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.1.13.2 | 0x59de7e7c, 0x078d, 0x4217, 0xa5, 0xfd, 0xf0, 0x1e, 0x15, 0xeb, 0xa3, 0x67 | EFI\_USB2\_HC\_PROTOCOL. ClearRootHubPortFeature - ClearRootHubPortFeature () returns EFI\_INVALID\_PARAMETER with an invalid PortFeature. | 1. Call GetRootHubPortNumber () to get the number of ports. The return status should be EFI\_SUCCESS.  2. Call ClearRootHubPortFeature() with a PortFeature not value of EfiUsbPortEnable, EfiUsbPortSuspend, EfiUsbPortPower, EfiUsbPortConnectChange, EfiUsbPortResetChange, EfiUsbPortEnableChange, EfiUsbPortSuspendChange, nor EfiUsbPortOverCurrentChange. The return status should be EFI\_INVALID\_PARAMETER. |

## EFI\_USB\_IO\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_USB\_IO\_PROTOCOL Section.

Most of functionalities rely on real USB devices. They are not covered in below checkpoints.

### UsbControlTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.1.1 | 0xe687694c, 0xc7ec, 0x444b, 0xac, 0xc5, 0xa3, 0x56, 0xf2, 0xb6, 0x3f, 0x15 | EFI\_USB\_IO\_PROTOCOL. UsbControlTransfer - UsbControlTransfer () returns EFI\_INVALID\_PARAMETER with an invalid TransferDirection. | 1. Call UsbControlTransfer() with an invalid TransferDirection ( value of -1). The return status should be EFI\_INVALID\_PARAMETER.  2. Call UsbControlTransfer() with an invalid TransferDirection ( value of 0x7FFFFFFF). The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.1.2 | 0x4aa535ad, 0x7985, 0x49f3, 0x81, 0x53, 0xa3, 0xd7, 0x04, 0x1e, 0x3f, 0xd0 | EFI\_USB\_IO\_PROTOCOL. UsbControlTransfer - UsbControlTransfer () returns EFI\_INVALID\_PARAMETER with a Request value of NULL. | 1. Call UsbControlTransfer() with a Request value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.1.3 | 0xc6bfebde, 0xd2d6, 0x44fa, 0xa6, 0xd9, 0x9b, 0x3c, 0x88, 0x9a, 0x52, 0x81 | EFI\_USB\_IO\_PROTOCOL. UsbControlTransfer - UsbControlTransfer () returns EFI\_INVALID\_PARAMETER with a Status value of NULL. | 1. Call UsbControlTransfer() with a Status value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.1.4 | �0x937f99d5, 0x18ef, 0x424c, 0xb4, 0x4c, 0x54, 0xaf, 0xf6, 0x20, 0xe0, 0xdc | EFI\_USB\_IO\_PROTOCOL.  UsbControlTransfer **-**  UsbControlTransfer () **returns**  EFI\_SUCCESS or EFI\_DEVICE\_ERROR when the parameter Timeout is 0. | 1. Call UsbControlTransfer () �when the parameter Timeout is 0. The return code must be  EFI\_SUCCESS or EFI\_DEVICE\_ERROR.  . |

### UsbBulkTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.2.1 | 0xf7c2276a, 0xfcd0, 0x4aeb, 0x99, 0x79, 0xf8, 0x79, 0x24, 0xd4, 0xc4, 0x83 | EFI\_USB\_IO\_PROTOCOL. UsbBulkTransfer - UsbBulkTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbBulkTransfer() with an invalid DeviceEndpoint value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.2.2 | 0xa0365348, 0xba4c, 0x43fe, 0xba, 0xde, 0x8e, 0x35, 0x26, 0x39, 0x7e, 0xbd | EFI\_USB\_IO\_PROTOCOL. UsbBulkTransfer - UsbBulkTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbBulkTransfer() with an invalid DeviceEndpoint ( value of 0x10). The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.2.3 | 0xafcf7b82, 0x16ad, 0x4721, 0x92, 0x46, 0x0d, 0x7b, 0xbb, 0xbd, 0xc9, 0x3a | EFI\_USB\_IO\_PROTOCOL. UsbBulkTransfer - UsbBulkTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbBulkTransfer() with an invalid DeviceEndpoint ( value of 0x80 ). The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.2.4 | 0x88c28425, 0xfbc6, 0x4441, 0x91, 0x23, 0x88, 0x83, 0x76, 0x9c, 0xed, 0x1e | EFI\_USB\_IO\_PROTOCOL. UsbBulkTransfer - UsbBulkTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbBulkTransfer() with an invalid DeviceEndpoint ( value of 0x90). The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.2.5 | 0x383c6bd1, 0xb1f3, 0x4987, 0x8c, 0x6f, 0xb5, 0xd5, 0x23, 0xb4, 0x93, 0xc1 | EFI\_USB\_IO\_PROTOCOL. UsbBulkTransfer - UsbBulkTransfer () returns EFI\_INVALID\_PARAMETER with a DeviceEndpoint value of not a BULK endpoint. | 1. Call UsbBulkTransfer() with an invalid DeviceEndpoint which is not a BULK endpoint. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.2.6 | 0x141aa66b, 0x7628, 0x4275, 0xae, 0xe3, 0x8c, 0xe1, 0x17, 0x65, 0x0d, 0xcc | EFI\_USB\_IO\_PROTOCOL. UsbBulkTransfer - UsbBulkTransfer () returns EFI\_INVALID\_PARAMETER with a Data value of NULL. | 1. Call UsbBulkTransfer() with a Data value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.2.7 | 0x486552a5, 0x9863, 0x4eed, 0x8b, 0x37, 0x92, 0xb3, 0x8b, 0xc3, 0xe3, 0xeb | EFI\_USB\_IO\_PROTOCOL. UsbBulkTransfer - UsbBulkTransfer () returns EFI\_INVALID\_PARAMETER with a DataLength value of NULL. | 1. Call UsbBulkTransfer() with a DataLength value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.2.8 | 0x582d809f, 0x88ce, 0x4a35, 0x89, 0xc6, 0xb5, 0x79, 0xf3, 0x70, 0x54, 0x66 | EFI\_USB\_IO\_PROTOCOL. UsbBulkTransfer - UsbBulkTransfer () returns EFI\_INVALID\_PARAMETER with a Status value of NULL. | 1. Call UsbBulkTransfer() with a Status value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.2.9 | 0x3d1b8608, 0x8c1e, 0x4b09, 0x81, 0x0f, 0xd9, 0x5c, 0x2a, 0xd7, 0x66, 0xae | EFI\_USB\_IO\_PROTOCOL.  UsbBulkTransfer -  UsbBulkTransfer () returns  EFI\_SUCCESS or EFI\_DEVICE\_ERROR when  the parameter Timeout is 0. | 1. Call UsbBulkTransfer () when the parameter Timeout is 0. The return code must be  EFI\_SUCCESS or  EFI\_DEVICE\_ERROR. |

### UsbAsyncInterruptTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.3.1 | 0x551fbef7, 0xd9e9, 0x4302, 0xa4, 0xcd, 0x2d, 0xb6, 0x83, 0x47, 0xc9, 0x4a | EFI\_USB\_IO\_PROTOCOL. UsbAsyncInterruptTransfer - UsbAsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbAsyncInterruptTransfer () with an invalid DeviceEndpoint value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.3.2 | 0xbb293ec7, 0x3a01, 0x493d, 0xa2, 0x2b, 0x71, 0x97, 0x48, 0x0b, 0x4f, 0x64 | EFI\_USB\_IO\_PROTOCOL. UsbAsyncInterruptTransfer - UsbAsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbAsyncInterruptTransfer () with an invalid DeviceEndpoint value of 0x10. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.3.3 | 0xf2436425, 0xee55, 0x41ee, 0x81, 0x3d, 0xa4, 0x64, 0x47, 0x17, 0x18, 0xfa | EFI\_USB\_IO\_PROTOCOL. UsbAsyncInterruptTransfer - UsbAsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbAsyncInterruptTransfer () with an invalid DeviceEndpoint**(** value of 0x80**)**. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.3.4 | 0x7ab9696d, 0x6687, 0x4f7f, 0xac, 0x16, 0x6a, 0x60, 0x23, 0x57, 0x41, 0xa7 | EFI\_USB\_IO\_PROTOCOL. UsbAsyncInterruptTransfer - UsbAsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbAsyncInterruptTransfer () with an invalid DeviceEndpoint ( value of 0x90). The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.3.5 | 0x17646b64, 0x413f, 0x41cc, 0xbd, 0x8c, 0x91, 0x66, 0xe4, 0xef, 0x3e, 0x4c | EFI\_USB\_IO\_PROTOCOL. UsbAsyncInterruptTransfer - UsbAsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a DeviceEndpoint value of not an Interrupt endpoint. | 1. Call UsbAsyncInterruptTransfer () with a DeviceEndpoint value of not an Interrupt endpoint. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.3.6 | 0x4d89db86, 0x4acc, 0x4ed8, 0xb8, 0xd1, 0xc3, 0xaa, 0x75, 0x08, 0xb3, 0xee | EFI\_USB\_IO\_PROTOCOL. UsbAsyncInterruptTransfer - UsbAsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid PollingInterval. | 1. Call UsbAsyncInterruptTransfer () with an invalid PollingInterval value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.3.7 | 0x808d9c7c, 0x2397, 0x406d, 0x97, 0x69, 0xcd, 0xeb, 0x4f, 0xde, 0x11, 0x16 | EFI\_USB\_IO\_PROTOCOL. UsbAsyncInterruptTransfer - UsbAsyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid PollingInterval. | 1. Call UsbAsyncInterruptTransfer () with an invalid PollingInterval (value of 256. The return status should be EFI\_INVALID\_PARAMETER. |

### UsbSyncInterruptTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.4.1 | 0x59735398, 0x5d31, 0x42e2, 0x8e, 0x65, 0x68, 0xbd, 0x6c, 0x1e, 0xbb, 0xb6 | EFI\_USB\_IO\_PROTOCOL. UsbSyncInterruptTransfer - UsbSyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbSyncInterruptTransfer () with an invalid DeviceEndpoint (value of 0). The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.4.2 | 0xdd2221a8, 0x7dc1, 0x4d2a, 0x85, 0x99, 0x6b, 0x86, 0x9d, 0x74, 0xf0, 0xa7 | EFI\_USB\_IO\_PROTOCOL. UsbSyncInterruptTransfer - UsbSyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbSyncInterruptTransfer () with an invalid DeviceEndpoint value of 0x10. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.4.3 | 0x15c6a9c5, 0x9912, 0x4474, 0xac, 0xe5, 0xa3, 0x1d, 0x49, 0xde, 0x63, 0x28 | EFI\_USB\_IO\_PROTOCOL. UsbSyncInterruptTransfer - UsbSyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbSyncInterruptTransfer () with an invalid DeviceEndpoint value of 0x80. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.4.4 | 0x833ca596, 0xf83d, 0x455f, 0x95, 0x95, 0xe5, 0x77, 0xa6, 0xaf, 0x62, 0xdc | EFI\_USB\_IO\_PROTOCOL. UsbSyncInterruptTransfer - UsbSyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbSyncInterruptTransfer () with an invalid DeviceEndpoint value of 0x90. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.4.5 | 0x60a2a3d0, 0xb657, 0x413d, 0x9b, 0x1c, 0xa7, 0x2b, 0x46, 0xaa, 0xa6, 0x77 | EFI\_USB\_IO\_PROTOCOL. UsbSyncInterruptTransfer - UsbSyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a DeviceEndpoint value of not an Interrupt endpoint. | 1. Call UsbSyncInterruptTransfer () with a DeviceEndpoint value of not an Interrupt endpoint. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.4.6 | 0xd4730bf3, 0x8b92, 0x4bcf, 0x99, 0xef, 0xe1, 0xdb, 0x65, 0xe9, 0x86, 0xec | EFI\_USB\_IO\_PROTOCOL. UsbSyncInterruptTransfer - UsbSyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a Data value of NULL. | 1. Call UsbSyncInterruptTransfer () with a Data value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.4.7 | 0x0dbc8bd6, 0x4405, 0x49c0, 0xa5, 0xd1, 0xbc, 0x01, 0xca, 0x61, 0x67, 0xb2 | EFI\_USB\_IO\_PROTOCOL. UsbSyncInterruptTransfer - UsbSyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a DataLength value of NULL. | 1. Call UsbSyncInterruptTransfer () with a DataLength value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.4.8 | 0xa5e94a41, 0xc3ef, 0x4172, 0x94, 0xc2, 0xc7, 0xba, 0xa8, 0x72, 0xc3, 0x74 | EFI\_USB\_IO\_PROTOCOL. UsbSyncInterruptTransfer - UsbSyncInterruptTransfer () returns EFI\_INVALID\_PARAMETER with a Status value of NULL. | 1. Call UsbSyncInterruptTransfer () with a Status value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |

### UsbIsochronousTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.5.1 | 0x006bb343, 0x842a, 0x417a, 0xa8, 0x23, 0x29, 0x75, 0x68, 0x9b, 0x9e, 0x2a | EFI\_USB\_IO\_PROTOCOL. UsbIsochronousTransfer - UsbIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbIsochronousTransfer () with an invalid DeviceEndpoint value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.5.2 | 0xd4f5400e, 0x3ed0, 0x4659, 0xa4, 0x80, 0xff, 0xf5, 0xeb, 0x8b, 0xae, 0x9b | EFI\_USB\_IO\_PROTOCOL. UsbIsochronousTransfer - UsbIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbIsochronousTransfer () with an invalid DeviceEndpoint value of 0x10. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.5.3 | 0xcfbc4d53, 0x07b7, 0x4366, 0x85, 0x98, 0x85, 0xf1, 0x6a, 0x15, 0x82, 0xb3 | EFI\_USB\_IO\_PROTOCOL. UsbIsochronousTransfer - UsbIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbIsochronousTransfer () with an invalid DeviceEndpoint value of 0x80. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.5.4 | 0xc9cc277e, 0x02a3, 0x4392, 0x82, 0x24, 0x87, 0xe5, 0x26, 0x21, 0xfd, 0xd6 | EFI\_USB\_IO\_PROTOCOL. UsbIsochronousTransfer - UsbIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbIsochronousTransfer () with an invalid DeviceEndpoint value of 0x90. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.5.5 | 0x686e7854, 0xe518, 0x41c1, 0xb1, 0x71, 0x60, 0x4e, 0x6f, 0x7e, 0xe2, 0x91 | EFI\_USB\_IO\_PROTOCOL. UsbIsochronousTransfer - UsbIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with a DeviceEndpoint which is not an Isochronous endpoint. | 1. Call UsbIsochronousTransfer () with a DeviceEndpoint value of not an Isochronous endpoint. The return status should be EFI\_INVALID\_PARAMETER. |

### UsbAsyncIsochronousTransfer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.6.1 | 0x5a8a2a48, 0xd6cc, 0x4993, 0x82, 0x1e, 0xf7, 0x2f, 0x48, 0x40, 0xa7, 0x26 | EFI\_USB\_IO\_PROTOCOL. UsbAsyncIsochronousTransfer - UsbAsyncIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbAsyncIsochronousTransfer () with an invalid DeviceEndpoint value of 0. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.6.2 | 0x7df33f6b, 0x7525, 0x4999, 0x83, 0x9c, 0xb2, 0xc7, 0x73, 0xd1, 0xa2, 0xa5 | EFI\_USB\_IO\_PROTOCOL. UsbAsyncIsochronousTransfer - UsbAsyncIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbAsyncIsochronousTransfer () with an invalid DeviceEndpoint value of 0x10. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.6.3 | 0x586d899f, 0x34f8, 0x474d, 0x99, 0x5e, 0x9e, 0x3e, 0x98, 0x9f, 0xf0, 0xee | EFI\_USB\_IO\_PROTOCOL. UsbAsyncIsochronousTransfer - UsbAsyncIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbAsyncIsochronousTransfer () with an invalid DeviceEndpoint value of 0x80. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.6.4 | 0xfbe98aec, 0xeab8, 0x45a3, 0x85, 0xd3, 0x00, 0x32, 0x0d, 0x1c, 0xaa, 0xe3 | EFI\_USB\_IO\_PROTOCOL. UsbAsyncIsochronousTransfer - UsbAsyncIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with an invalid DeviceEndpoint. | 1. Call UsbAsyncIsochronousTransfer () with an invalid DeviceEndpoint value of 0x90. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.6.5 | 0x7588b124, 0xdaa7, 0x4715, 0xa1, 0x99, 0xa4, 0xdc, 0x32, 0x19, 0x1c, 0xc9 | EFI\_USB\_IO\_PROTOCOL. UsbAsyncIsochronousTransfer - UsbAsyncIsochronousTransfer () returns EFI\_INVALID\_PARAMETER with a DeviceEndpoint value of not an Isochronous endpoint. | 1. Call UsbAsyncIsochronousTransfer () with a DeviceEndpoint value of not an Isochronous endpoint. The return status should be EFI\_INVALID\_PARAMETER. |

### UsbGetDeviceDescriptor()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.7.1 | 0xe789ba3f, 0x2405, 0x4d45, 0xbf, 0xdb, 0x7e, 0xa7, 0xe8, 0x33, 0xc6, 0x8b | EFI\_USB\_IO\_PROTOCOL. UsbGetDeviceDescriptor - UsbGetDeviceDescriptor () returns EFI\_INVALID\_PARAMETER with a DeviceDescriptor value of NULL. | 1. Call UsbGetDeviceDescriptor () with a DeviceDescriptor value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |

### UsbGetConfigDescriptor()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.8.1 | 0x387570c3, 0x6923, 0x4cbb, 0x82, 0xb2, 0x59, 0xc7, 0x41, 0xab, 0x92, 0x4b | EFI\_USB\_IO\_PROTOCOL. UsbGetConfigDescriptor - UsbGetConfigDescriptor () returns EFI\_INVALID\_PARAMETER with a ConfigurationDescriptor value of NULL. | 1. Call UsbGetConfigDescriptor () with a ConfigurationDescriptor value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |

### UsbGetInterfaceDescriptor()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.9.1 | 0x47c33713, 0x8fbc, 0x43a4, 0xa2, 0xcd, 0xc1, 0x6b, 0xc7, 0xa5, 0xd4, 0x37 | EFI\_USB\_IO\_PROTOCOL. UsbGetInterfaceDescriptor - UsbGetInterfaceDescriptor () returns EFI\_INVALID\_PARAMETER with a InterfaceDescriptor value of NULL. | 1. Call UsbGetInterfaceDescriptor () with a InterfaceDescriptor value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |

### UsbGetEndpointDescriptor()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.10.1 | 0x8167f778, 0xa58c, 0x4837, 0xaf, 0xfb, 0x5e, 0x10, 0x69, 0x66, 0xa8, 0x74 | EFI\_USB\_IO\_PROTOCOL. UsbGetEndpointDescriptor - UsbGetEndpointDescriptor () returns EFI\_INVALID\_PARAMETER with an EndpointDescriptor value of NULL. | 1. Call UsbGetEndpointDescriptor () with an EndpointDescriptor value of NULL. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.10.2 | 0xb0da5669, 0x163d, 0x4d93, 0xae, 0xf0, 0x7b, 0x28, 0x53, 0x5f, 0x47, 0x3e | EFI\_USB\_IO\_PROTOCOL. UsbGetEndpointDescriptor - UsbGetEndpointDescriptor () returns EFI\_INVALID\_PARAMETER with an EndpointIndex value of larger than 15. | 1. Call UsbGetEndpointDescriptor () with an EndpointIndex value of larger than 15. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.21.2.10.3 | 0x692ec6a6, 0x057d, 0x43c3, 0x94, 0x74, 0x5c, 0x29, 0xb2, 0x5e, 0x5c, 0xe5 | EFI\_USB\_IO\_PROTOCOL. UsbGetEndpointDescriptor - UsbGetEndpointDescriptor () returns EFI\_INVALID\_PARAMETER with an EndpointIndex value of equal to the number of endpoints. | 1. Call UsbGetInterfaceDescriptor () to get the number of endpoints. The return status should be EFI\_SUCCESS.  2. Call UsbGetEndpointDescriptor () with an EndpointIndex value of equal to the number of endpoints. The return status should be EFI\_INVALID\_PARAMETER. |

### UsbPortReset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.21.2.11.1 | 0x27431330, 0x54c8, 0x40fe, 0x93, 0x74, 0x9d, 0x39, 0x4d, 0x10, 0x75, 0x3b | EFI\_USB\_IO\_PROTOCOL. UsbPortReset - UsbPortReset () returns EFI\_INVALID\_PARAMETER with a USB hub. | 1. Call UsbPortReset () with a USB hub. The return status should be EFI\_INVALID\_PARAMETER. |