TABLE OF CONTENTS

TABLE OF CONTENTS ....................................................................................................................... 2
MODIFICATION RECORD................................................................................................................. 5
ACKNOWLEDGMENTS ...................................................................................................................... 6
INTRODUCTION ................................................................................................................................... 7
REFERENCES........................................................................................................................................ 9
ADDITIONAL ACRONYMS AND ABBREVIATIONS .................................................................. 10
TEST SETUPS ...................................................................................................................................... 11

GROUP 1: Basic Multiconnection Tests for Targets .......................................................... 12
  Test #1.1.1 Basic READ with 2 connections ...................................................................................... 13
  Test #1.1.2 Basic READ with 4 connections ...................................................................................... 14
  Test #1.2.1 Basic WRITE with 2 connections .................................................................................... 15
  Test #1.2.2 Basic WRITE with 4 connections .................................................................................... 16
  Test #1.3.1: Connection Allegiance .................................................................................................... 17

GROUP 2: Parameter Negotiation Multiconnection Tests for Targets ....................... 18
  Test #2.1.1 MaxConnections Key (Informative) ................................................................................ 19
  Test #2.1.2 MaxConnections Renegotiation Check ............................................................................ 20
  Test #2.1.3 MaxConnections Renegotiation Attempt ......................................................................... 21
  Test #2.2.1 TargetName on Two Connections ................................................................................... 22
  Test #2.2.2 Missing TargetName on All Connections ........................................................................ 23
  Test #2.2.3 Missing TargetName on Second Connection Only .......................................................... 24
  Test #2.3.1 InitiatorName on Two Connections ............................................................................... 25
  Test #2.3.2 InitiatorName Redeclared on Second Connection ........................................................... 26
  Test #2.4.1 TargetAlias Declaration ................................................................................................ 27
  Test #2.5.1 InitialR2T=Yes Across Two Connections ...................................................................... 28
  Test #2.5.2 InitialR2T=No Across Two Connections ...................................................................... 29
  Test #2.5.3 InitialR2T=Yes Renegotiation Attempt with Same Value ................................................ 30
  Test #2.5.4 InitialR2T=Yes Renegotiation Attempt with InitialR2T=No .......................................... 31
  Test #2.5.5 InitialR2T=No Renegotiation Attempt with Same Value ................................................ 32
  Test #2.5.6 InitialR2T=No Renegotiation Attempt with InitialR2T=Yes .......................................... 33
  Test #2.6.1 ImmediateData=Yes Across Two Connections ............................................................... 34
  Test #2.6.2 ImmediateData=No Across Two Connections ................................................................ 35
  Test #2.6.3 ImmediateData=Yes Renegotiation Attempt with Same Value ..................................... 36
  Test #2.6.4 ImmediateData=Yes Renegotiation Attempt with ImmediateData=No ........................ 37
  Test #2.6.5 ImmediateData=No Renegotiation Attempt with Same Value ..................................... 38
  Test #2.6.6 ImmediateData=No Renegotiation Attempt with ImmediateData=Yes ........................ 39
  Test #2.7.1 MaxRecvDataSegmentLength Declared on First Connection Only .............................. 40
  Test #2.7.2 MaxRecvDataSegmentLength Declared on Second Connection Only ........................... 41
  Test #2.7.3 MaxRecvDataSegmentLength Declared with Larger Value on Second Connection .... 42
  Test #2.7.4 MaxRecvDataSegmentLength Declared with Smaller Value on Second Connection ..... 43
The University of New Hampshire
InterOperability Laboratory

Test #2.8.1 MaxBurstLength Key.................................................................44
Test #2.8.2 MaxBurstLength Renegotiation Attempt.................................45
Test #2.9.1 FirstBurstLength Key..............................................................46
Test #2.9.2 FirstBurstLength Renegotiation Attempt.................................47
Test #2.10.1 DefaultTime2Wait Renegotiation Check.................................48
Test #2.10.2 DefaultTime2Wait Renegotiation Attempt.................................49
Test #2.11.1 DefaultTime2Retain Renegotiation Check...............................50
Test #2.11.2 DefaultTime2Retain Renegotiation Attempt.............................51
Test #2.12.1 TaskReporting Renegotiation Check.........................................52
Test #2.12.2 TaskReporting Renegotiation Attempt......................................53
Test #2.13.1 MaxOutstandingR2T Renegotiation Check...............................54
Test #2.13.2 MaxOutstandingR2T Renegotiation Attempt.............................55
Test #2.13.3 MaxOutstandingR2T Key.........................................................56
Test #2.14.1 DataPDUInOrder=Yes Across Two Connections......................57
Test #2.14.2.1 DataPDUInOrder=No Across Two Connections With InitialR2T=Yes......58
Test #2.14.2.2 DataPDUInOrder=No Across Two Connections with InitialR2T=No......59
Test #2.14.3 DataPDUInOrder=Yes Renegotiation Attempt with Same Value.........60
Test #2.14.4 DataPDUInOrder=Yes Renegotiation Attempt with DataPDUInOrder=No....61
Test #2.14.5 DataPDUInOrder=No Renegotiation Attempt with Same Value........62
Test #2.14.6 DataPDUInOrder=No Renegotiation Attempt with DataPDUInOrder=Yes....63
Test #2.15.1 READ Command when DataSequenceInOrder=Yes Across Two Connections 64
Test #2.15.2 WRITE Command when DataSequenceInOrder=Yes Across Two Connections 65
Test #2.15.3 DataSequenceInOrder=Yes Renegotiation Attempt with Same Value......66
Test #2.15.4 DataSequenceInOrder=Yes Renegotiation Attempt with DataSequenceInOrder=Yes 67
Test #2.15.5 DataSequenceInOrder=No Renegotiation Attempt with Same Value......68
Test #2.15.6 DataSequenceInOrder=No Renegotiation Attempt with DataSequenceInOrder=Yes 69
Test #2.16.1 HeaderDigest and DataDigest Keys Across Two Connections........70
Test #2.16.2 HeaderDigest and DataDigest Keys Across Three Connections.........71

GROUP 3: Sequence Number Multiconnection Tests for Targets....................72
Test #3.1.1 In Order CmdSN Wraparound Across Two Connections..................73
Test #3.1.2 Out of Order CmdSN Wraparound Across Two Connections...............74
Test #3.1.3 In Order CmdSN Wraparound on One Connection..........................75
Test #3.2.1: Duplicate CmdSN with Two Connections (First Command in Session)......76
Test #3.2.2: Duplicate CmdSN with Two Connections (Not First Command in Session) 77
Test #3.2.3 Out of Range CmdSN with Two Connections..................................78
Test #3.3.1 StatSN Ordering with Two Connections.......................................79
Test #3.4.1 DataSN Ordering with Two Connections.......................................80
Test #3.5.1 R2TSN Ordering with Two Connections........................................81

GROUP 4: Connection Termination Multiconnection Tests for Targets.............82
Test #4.1.1 Task Reassignment Attempt before DefaultTime2Wait Timer Expires...83
Test #4.1.2 Task Reassignment Attempt after DefaultTime2Retain Timer Expires....84
Test #4.1.3 Task Reassignment within DefaultTime2Wait and DefaultTime2Retain Range 85
Test #4.2.1: Logout Response Close Existent Connection..............................86
Test #4.2.2: Logout Response Close Session................................................87
Test #4.2.3: Logout Response Close Nonexistent Connection........................88

GROUP 5: Task Management Multiconnection Tests for Targets......................89
Test #5.1.1: Clarified RFC3720 Multi-Task Abort Semantics..........................90
<table>
<thead>
<tr>
<th>Test #</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.1</td>
<td>Response Fence Check Prior to Task Management Function Response</td>
<td>91</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Response Fence Check After Task Management Function Response</td>
<td>93</td>
</tr>
<tr>
<td>5.3.1</td>
<td>FastAbort when Affected Tasks Have Not Been Received</td>
<td>94</td>
</tr>
<tr>
<td>5.3.2</td>
<td>FastAbort with Active Affected Target Transfer Tags</td>
<td>95</td>
</tr>
<tr>
<td>5.3.3</td>
<td>FastAbort Asynchronous Message Check</td>
<td>96</td>
</tr>
<tr>
<td>5.3.4</td>
<td>FastAbort Multiple LUNs</td>
<td>97</td>
</tr>
<tr>
<td>5.3.5</td>
<td>FastAbort Response Fence Check</td>
<td>98</td>
</tr>
<tr>
<td>5.4.1</td>
<td>FastAbort Logical Unit Reset with FastAbort Third-Party Session</td>
<td>99</td>
</tr>
<tr>
<td>5.4.2</td>
<td>FastAbort Logical Unit Reset with RFC3720 Third-Party Session</td>
<td>102</td>
</tr>
<tr>
<td>5.4.3</td>
<td>RFC3720 Logical Unit Reset with FastAbort Third-Party Session</td>
<td>105</td>
</tr>
</tbody>
</table>
MODIFICATION RECORD

[1] December 6, 2004 (Version 0.2) DRAFT RELEASE
    David Woolf: Initial draft release

    David Woolf: Test Suite updated to match final RFC 3720 standard.

    Patrick MacArthur: Changed most of the test labels so that each is unique
    Grouped and renumbered tests:

    | From     | To                  |
    |----------|---------------------|
    | #1.1-1.2 | #1.1.1, 1.1.2       |
    | #1.3-1.4 | #1.2.1, 1.2.2       |
    | #2.1-12.3 | #2.1.1-2.11.3      |
    | #14.1-17.3 | #2.13.1-2.16.3     |
    | #18.1-18.3 | #3.1.1-3.1.3      |
    | #18.4    | #3.2.3              |
    | #19.1-21.1 | #3.3.1-3.5.1      |
    | #13.1-13.3 | #4.1.1-4.1.3      |

    Added test #1.3 from Target FFP test suite v1.2 test #6.1
    Added tests #4.2.* from Target FFP test suite v1.2 tests #1.3.*
    Added test #5.1.1 from Target FFP test suite v1.2 test #11.3.2
    Added tests #3.2.1 from Target FFP test suite v1.3 test #1.3.2
    Added tests #3.2.2 from Target FFP test suite v1.3 test #1.3.4
    Modified test #3.2.3 to remove redundancy
    Updated test suite to match RFC 5048:
      Added tests #1.12.1 and #1.12.2
      Added tests #3.9.*, #3.10.1, and #3.10.2
      Added tests #5.2.*, #5.3.*, and #5.4.*
ACKNOWLEDGMENTS

The University of New Hampshire would like to acknowledge the efforts of the following individuals in the development of this test suite.

David Woolf     University of New Hampshire
Ethan Burns     University of New Hampshire
Patrick MacArthur University of New Hampshire
INTRODUCTION

The University of New Hampshire’s InterOperability Laboratory (IOL) is an institution designed to improve the interoperability of standards based products by providing an environment where a product can be tested against other implementations of a standard. This particular suite of tests has been developed to help implementers evaluate the Multiconnection functionality of their iSCSI targets.

These tests are designed to determine if an iSCSI product conforms to specifications defined in both IETF RFC 3720 iSCSI (hereafter referred to as the “iSCSI Standard”) as well as updates as contained in IETF RFC 5048 iSCSI Corrections and Clarifications RFC (hereafter referred to as “iSCSI Corrections and Clarifications”). Successful completion of all tests contained in this suite does not guarantee that the tested device will successfully operate with other iSCSI products. However, when combined with satisfactory operation in the IOL’s interoperability test bed, these tests provide a reasonable level of confidence that the Device Under Test (DUT) will function properly in many iSCSI environments.

The tests contained in this document are organized in order to simplify the identification of information related to a test, and to facilitate the actual testing process. Tests are separated into groups, primarily in order to reduce setup time in the lab environment, however the different groups typically also tend to focus on specific aspects of device functionality. A dot-notated naming system is used to catalog the tests, where the first number always indicates a specific group of tests in which the test suite is based. The second and third numbers indicate the test’s group number and test number within that group, respectively. This format allows for the addition of future tests in the appropriate groups without requiring the renumbering of the subsequent tests.

The test definitions themselves are intended to provide a high-level description of the motivation, resources, procedures, and methodologies specific to each test. Formally, each test description contains the following sections:

Purpose
The purpose is a brief statement outlining what the test attempts to achieve. The test is written at the functional level.

References
This section specifies all reference material external to the test suite, including the specific sub clauses references for the test in question, and any other references that might be helpful in understanding the test methodology and/or test results. External sources may also be referenced by a bracketed number (e.g., [1]) when mentioned in the test description. Any other references in the test description that are not indicated in this manner refer to elements within the test suite document itself (e.g., “Appendix 5.A”, or “Table 5.1.1-1”)

Resource Requirements
The requirements section specifies the test hardware and/or software needed to perform the test. This is generally expressed in terms of minimum requirements, however in some cases specific equipment manufacturer/model information may be provided.
**Last Modification**
This specifies the date of the last modification to this test.

**Discussion**
The discussion covers the assumptions made in the design or implementation of the test, as well as known limitations. Other items specific to the test are covered here as well.

**Test Setup**
The setup section describes the initial configuration of the test environment. Small changes in the configuration should not be included here, and are generally covered in the test procedure section (next).

**Procedure**
The procedure section of the test description contains the systematic instructions for carrying out the test. It provides a cookbook approach to testing, and may be interspersed with observable results.

**Observable Results**
This section lists the specific observables that can be examined by the tester in order to verify that the DUT is operating properly. When multiple values for an observable are possible, this section provides a short discussion on how to interpret them. The determination of a pass or fail outcome for a particular test is generally based on the successful (or unsuccessful) detection of a specific observable.

**Possible Problems**
This section contains a description of known issues with the test procedure, which may affect test results in certain situations. It may also refer the reader to test suite appendices and/or other external sources that may provide more detail regarding these issues.
REFERENCES

The following documents are referenced in this text:

iSCSI Standard IETF RFC 3720
iSCSI Corrections and Clarifications IETF RFC 5048
ADDITIONAL ACRONYMS AND ABBREVIATIONS

The acronyms and abbreviations defined here supplement the acronyms defined in IETF RFC 3720 section 2.2 and may be used in this document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUT</td>
<td>Device Under Test</td>
</tr>
<tr>
<td>DDTL</td>
<td>DesiredDataTransferLength</td>
</tr>
<tr>
<td>DSL</td>
<td>DataSegmentLength</td>
</tr>
<tr>
<td>EDTL</td>
<td>ExpectedDataTransferLength</td>
</tr>
<tr>
<td>MRDSL</td>
<td>MaxRecvDataSegmentLength</td>
</tr>
<tr>
<td>READ CAP</td>
<td>READ CAPACITY</td>
</tr>
<tr>
<td>TMF</td>
<td>Task Management Function</td>
</tr>
</tbody>
</table>
The University of New Hampshire
InterOperability Laboratory

TEST SETUPS

The following test setups are used in this test suite:

Test Setup 1:

![Diagram of Test Setup 1 showing TCP Connection(s) between Testing Station/Monitor and DUT]
The University of New Hampshire
InterOperability Laboratory

GROUP 1: Basic Multiconnection Tests for Targets

Overview: This group of tests verifies the basic functionality of Multiconnection specifications of iSCSI defined in RFC 3720 and RFC 5048. Comments and questions regarding the implementation of these tests are welcome, and may be forwarded to Peter Scruton, UNH InterOperability Lab (pjs@iol.unh.edu).
Test #1.1.1 Basic READ with 2 connections

**Purpose:** To verify that a device using multiple connections is able to complete SCSI commands on the open connections and properly sets the sequence numbers of each transmitted PDU.

**Reference:** iSCSI Standard Clause 3.2.2.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 19, 2007

**Discussion:** iSCSI performs ordered command delivery within a session. All commands (initiator-to-target PDUs) in transit from the initiator to the target are numbered. The command number is carried by the iSCSI PDU as CmdSN (Command-Sequence-Number). The numbering is session-wide. Command numbering starts with the first login request on the first connection of a session (the leading login on the leading connection) and command numbers are incremented by 1 for every non-immediate command issued afterward. On any connection, the iSCSI initiator MUST send the commands in increasing order of CmdSN, except for commands that are retransmitted due to digest error recovery and connection recovery.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Start a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=2.
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second connection of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On each connection, perform a READ operation.

**Observable Results:**
- Verify that MaxCmdSN and ExpCmdSN follow the CmdSN used by the Testing Station, and that these are kept as Session-wide counters. ExpCmdSN should be incremented for each command received in the Session.
- Verify that SCSI traffic appears on each connection, and that each SCSI command completes successfully.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
### Test #1.1.2 Basic READ with 4 connections

**Purpose:** To verify that a device using multiple connections is able to complete SCSI commands on the open connections and properly sets the sequence numbers of each transmitted PDU.

**Reference:** iSCSI Standard Clause 3.2.2.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 19, 2007

**Discussion:** iSCSI performs ordered command delivery within a session. All commands (initiator-to-target PDUs) in transit from the initiator to the target are numbered. The command number is carried by the iSCSI PDU as CmdSN (Command-Sequence-Number). The numbering is session-wide. Command numbering starts with the first login request on the first connection of a session (the leading login on the leading connection) and command numbers are incremented by 1 for every non-immediate command issued afterward. On any connection, the iSCSI initiator MUST send the commands in increasing order of CmdSN, except for commands that are retransmitted due to digest error recovery and connection recovery.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Start a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=4.
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second, third and fourth connections of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On each connection, perform a READ operation.

**Observable Results:**
- Verify that MaxCmdSN and ExpCmdSN follow the CmdSN used by the Testing Station, and that these are kept as Session-wide counters. ExpCmdSN should be incremented for each command received in the Session.
- Verify that SCSI traffic appears on each connection, and that each SCSI command completes successfully.

**Possible Problems:** If the device does not support more than 3 connections this item is not testable.
Test #1.2.1 Basic WRITE with 2 connections

**Purpose:** To verify that a device using multiple connections is able to complete SCSI commands on the open connections and properly sets the sequence numbers of each transmitted PDU.

**Reference:** iSCSI Standard Clause 3.2.2.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 26, 2007

**Discussion:** iSCSI performs ordered command delivery within a session. All commands (initiator-to-target PDUs) in transit from the initiator to the target are numbered. The command number is carried by the iSCSI PDU as CmdSN (Command-Sequence-Number). The numbering is session-wide. Command numbering starts with the first login request on the first connection of a session (the leading login on the leading connection) and command numbers are incremented by 1 for every non-immediate command issued afterward. On any connection, the iSCSI initiator MUST send the commands in increasing order of CmdSN, except for commands that are retransmitted due to digest error recovery and connection recovery.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Start a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=2.
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second connection of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On each connection, perform a WRITE operation.

**Observable Results:**
- Verify that MaxCmdSN and ExpCmdSN follow the CmdSN used by the Testing Station, and that these are kept as Session-wide counters. ExpCmdSN should be incremented for each command received in the Session.
- Verify that SCSI traffic appears on each connection, and that each SCSI command completes successfully.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #1.2.2 Basic WRITE with 4 connections

**Purpose:** To verify that a device using multiple connections is able to complete SCSI commands on the open connections and properly sets the sequence numbers of each transmitted PDU.

**Reference:** iSCSI Standard Clause 3.2.2.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 26, 2007

**Discussion:** iSCSI performs ordered command delivery within a session. All commands (initiator-to-target PDUs) in transit from the initiator to the target are numbered. The command number is carried by the iSCSI PDU as CmdSN (Command-Sequence-Number). The numbering is session-wide. Command numbering starts with the first login request on the first connection of a session (the leading login on the leading connection) and command numbers are incremented by 1 for every non-immediate command issued afterward. On any connection, the iSCSI initiator MUST send the commands in increasing order of CmdSN, except for commands that are retransmitted due to digest error recovery and connection recovery.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Start a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=4.
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second, third and fourth connections of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On each connection, perform a WRITE operation.

** Observable Results:**
- Verify that MaxCmdSN and ExpCmdSN follow the CmdSN used by the Testing Station, and that these are kept as Session-wide counters. ExpCmdSN should be incremented for each command received in the Session.
- Verify that SCSI traffic appears on each connection, and that each SCSI command completes successfully.

**Possible Problems:** If the device does not support more than 3 connections this item is not testable.
Test #1.3.1: Connection Allegiance

**Purpose:** To verify that an iSCSI target transmits data and status response PDUs on the same connection as the request PDU was received.

**Reference:** iSCSI Standard Clause 3.2.4.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** March 16, 2005

**Discussion:** For any iSCSI request issued over a TCP connection, the corresponding response and/or other related PDU(s) MUST be sent over the same connection. We call this "connection allegiance". If the original connection fails before the command is completed, the connection allegiance of the command may be explicitly reassigned to a different transport connection. Thus, if an initiator issues a READ command, the target MUST send the requested data, if any, followed by the status to the initiator over the same TCP connection that was used to deliver the SCSI command. If an initiator issues a WRITE command, the initiator MUST send the data, if any, for that command over the same TCP connection that was used to deliver the SCSI command. The target MUST return Ready To Transfer (R2T), if any, and the status over the same TCP connection that was used to deliver the SCSI command. Retransmission requests (SNACK PDUs) and the data and status that they generate MUST also use the same connection.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Start two connections from the Testing Station to the iSCSI target being tested.
- On each connection perform a standard login and proceed to the Full Feature Phase.
- On each connection issue a SCSI-INQUIRY to the DUT, wait for response data and status.
- On each connection issue a TEST-UNIT-READY to the DUT, wait for response.
- On each connection issue a READ-CAP to the DUT, wait for response data and status.
- On each connection issue a WRITE command to the DUT. Wait for the DUT to solicit data with R2T.
- On each connection issue a READ command to the DUT. Wait for Data-in PDUs from the DUT.

**Observable Results:**
- Verify that for each SCSI request transmitted by the Testing Station, the DUT transmitted response, Data-in, and R2T PDUs on the same connection.

**Possible Problems:** None.
Overview: This group of tests verifies the Login Phase and key negotiation functionality of multiconnection iSCSI sessions as defined in RFC 3720 and RFC 5048. Comments and questions regarding the implementation of these tests are welcome, and may be forwarded to Peter Scruton, UNH InterOperability Lab (pjs@iol.unh.edu).
Test #2.1.1 MaxConnections Key (Informative)

**Purpose:** To verify that a device negotiating a value for MaxConnections adheres to that value. This test is informative only.

**Reference:** iSCSI Standard Clause 12.2 and 3.2.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 23, 2009

**Discussion:** Initiator and Target negotiate the maximum number of connections requested/acceptable. MaxConnections is a Session-wide parameter that can only be transmitted in the Leading connection of a session, and should not appear in any subsequent connections.

iSCSI targets and initiators MUST support at least one TCP connection and MAY support several connections in a session. For error recovery purposes, targets and initiators that support a single active connection in a session SHOULD support two connections during recovery.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- In the leading login of a session, negotiate a value of 1 for MaxConnections. This test is only feasible if it is already known that the DUT can support more than 1 connection.
- After MaxConnections has been negotiated, the Testing Station should attempt to open an 'extra' connection with the DUT, one more than the value of MaxConnections. It is expected that the DUT accepts this connection.
- After the login on the second connection is complete, the Testing Station should open a third connection. It is expected that this connection be refused by the target or reject with Login Reject reason of 'Too Many connections'.

**Observable Results:**
- Verify that the DUT does not accept any more connections from the Testing Station than 1 more than the value allowed by MaxConnections, though it may support even more connections.
- If the ErrorRecoveryLevel is 2, the DUT must accept the second connection and should reject the third connection. If the ErrorRecoveryLevel is less than 2, the DUT should accept the second connection and should reject the third connection, although rejecting both the second and third connection is acceptable in this case.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.1.2 MaxConnections Renegotiation Check

**Purpose:** To verify that a device adheres to a value for MaxConnections.

**Reference:** iSCSI Standard Clause 12.2

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 26, 2007

**Discussion:** Initiator and Target negotiate the maximum number of connections requested/acceptable. MaxConnections is a Session-wide parameter that can only be transmitted in the Leading connection of a session, and should not appear in any subsequent connections.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Negotiate a value of 4 for MaxConnections.
- Open 4 connections with the DUT.

**Observable Results:**
- Verify that in the second connection the DUT does not attempt to re-negotiate the MaxConnections parameter.

**Possible Problems:** If the device does not support more than 1 connection.
Test #2.1.3 MaxConnections Renegotiation Attempt

Purpose: To verify that a device negotiating a value for MaxConnections adheres to that value.

Reference: iSCSI Standard Clause 5.3, 12.2

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 26, 2007

Discussion: Initiator and Target negotiate the maximum number of connections requested/acceptable. An attempt to renegotiate/redeclare parameters not specifically allowed MUST be detected by the initiator and target. If such an attempt is detected by the target, the target MUST respond with Login reject (initiator error); if detected by the initiator, the initiator MUST drop the connection.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Negotiate a value of 2 or more for MaxConnections.
- Open a second connection with the DUT.
- On the second connection, offer a value of 4 for MaxConnections. This should appear as a re-negotiation of the MaxConnections key to the DUT.

Observable Results:
- Verify that in the second connection the DUT recognizes the re-negotiation and responds with Login Reject.

Possible Problems: If the device does not support more than 1 connection.
The University of New Hampshire
InterOperability Laboratory

Test #2.2.1 TargetName on Two Connections

**Purpose:** To verify that an iSCSI Target uses the TargetName on every connection within a session.

**Reference:** iSCSI Standard Clause 5.3, 12.4

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 26, 2007

**Discussion:** An iSCSI target is identified externally by a TargetName. For any connection within a session of SessionType="Normal" the first login request MUST have the TargetName key=value pair. TargetName MUST not be redeclared within the login phase. When a Target detects a redeclaration it should transmit a Login Reject.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Start a new session of SessionType="Normal" from the Testing Station to the DUT.
- In this new "Normal" session, negotiate a value for MaxConnections greater than 1.
- The Testing Station should open a second connection within this Normal session with the DUT.
- The Testing Station should declare the same TargetName in the first login request on each connection.

**Observable Results:**
- Verify that the DUT does not recognize this as a renegotiation of the TargetName key and continues with the Login Phase.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.2.2 Missing TargetName on All Connections

Purpose: To verify that an iSCSI Target uses the TargetName on every connection within a session.

Reference: iSCSI Standard Clause 5.3, 12.4

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: An iSCSI target is identified externally by a TargetName. For any connection within a session of SessionType="Normal" the first login request MUST have the TargetName key=value pair.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Start a new session of SessionType="Normal" from the Testing Station to the DUT.
- In this new "Normal" session, negotiate a value for MaxConnections greater than 1.
- The Testing Station should open a second connection within this Normal session with the DUT.
- The Testing Station should not declare the TargetName in either of the open connections.

Observable Results:
- Verify that the DUT recognizes the TargetName key was not provided and responds with Login Reject with Status Class and Status Detail of 0207 Missing Parameter.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.2.3 Missing TargetName on Second Connection Only

Purpose: To verify that an iSCSI Target uses the TargetName on every connection within a session.

Reference: iSCSI Standard Clause 5.3, 12.4

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: An iSCSI target is identified externally by a TargetName. For any connection within a session of SessionType="Normal" the first login request MUST have the TargetName key=value pair.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Start a new session of SessionType="Normal" from the Testing Station to the DUT.
- In this session, negotiate a value for MaxConnections greater than 1.
- The Testing Station should open a second connection within this Normal session with the DUT.
- The Testing Station should declare the TargetName in the first Login Request of the first connection only.

Observable Results:
- Verify that the DUT recognizes the TargetName key was properly provided on one connection.
- Verify that the DUT recognizes that it did not receive the TargetName key on the second connection and transmits Login Reject with status 'Missing Parameter'.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.3.1 InitiatorName on Two Connections

**Purpose:** To verify that an iSCSI target checks for the use of InitiatorName on every connection within a session.

**Reference:** iSCSI Standard Clause 12.5

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 27, 2007

**Discussion:** The initiator of the TCP connection MUST provide this key to the remote endpoint at the first Login of the Login Phase for every connection. The Initiator key enables the initiator to identify itself to the remote endpoint. InitiatorName MUST not be redeclared within the login phase.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate a value for MaxConnections greater than 1.
- Open a new connection within the "Normal" session.
- On each connection, the Testing Station should declare the InitiatorName key in the first Login Request.

**Observable Results:**
- Verify that the DUT accepts the InitiatorName in each connection, not interpreting this as a renegotiation despite InitiatorName being a Session-wide key.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.3.2 InitiatorName Redeclared on Second Connection

**Purpose:** To verify that an iSCSI initiator uses an InitiatorName uses it on every connection within a session.

**Reference:** iSCSI Standard Clause 12.5

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 27, 2007

**Discussion:** The initiator of the TCP connection MUST provide this key to the remote endpoint at the first Login of the Login Phase for every connection. The Initiator key enables the initiator to identify itself to the remote endpoint. InitiatorName MUST not be redeclared within the login phase.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate a value for MaxConnections greater than 1.
- Open a new connection, within the "Normal" session.
- On each connection, offer the InitiatorName key.
- On the second connection, offer the InitiatorName key twice.

**Observable Results:**
- Verify that the DUT recognizes the redeclaration of the InitiatorName key on the second connection and transmits a Login Reject.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.4.1 TargetAlias Declaration

Purpose: To verify that an iSCSI target uses a TargetAlias properly.

Reference: iSCSI Standard Clause 12.6

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 27, 2007

Discussion: If a target has been configured with a human-readable name or description, this name SHOULD be communicated to the initiator during a Login Response PDU if SessionType=Normal (see Section 12.21 SessionType). This string is not used as an identifier, nor is it meant to be used for authentication or authorization decisions. It can be displayed by the initiator's user interface in a list of targets to which it is connected.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading connection of the session, negotiate a value for MaxConnections greater than 1.
- Allow the DUT to open a new connection within the "Normal" session.

Observable Results:
- Verify that the DUT includes the TargetAlias key in a Login Response PDU if the DUT was configured with a TargetAlias.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.5.1 InitialR2T=Yes Across Two Connections

Purpose: To verify that an iSCSI target negotiates the InitialR2T key properly when using multiple connections.

Reference: iSCSI Standard Clause 12.10

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 27, 2007

Discussion: The InitialR2T key is used to toggle the use of R2T for the first outgoing data burst between initiator and target. InitialR2T is defined as a Session-wide parameter.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate InitialR2T=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection complete the Login Phase, and proceed to Full Feature Phase.
- On each connection perform a WRITE command.

Observable Results:
- Verify the DUT does not offer the InitialR2T key on the second connection.
- Verify that on each connection the DUT transmits an R2T before receiving any data associated with the WRITE command.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.5.2 InitialR2T=No Across Two Connections

**Purpose:** To verify that an iSCSI target negotiates the InitialR2T key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 12.10

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** February 16, 2009

**Discussion:** The InitialR2T key is used to toggle the use of R2T for the first outgoing data burst between initiator and target. InitialR2T is defined as a Session-wide parameter.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate InitialR2T=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, complete the Login Phase, and proceed to Full Feature Phase.
- On each connection perform a WRITE command.

**Observable Results:**
- Verify the DUT does not offer the InitialR2T key on the second connection.
- Verify that on each connection the DUT does not transmit R2T before receiving any data associated with the WRITE command.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable. If the DUT does not support InitialR2T=No this item is not testable.
Test #2.5.3 InitialR2T=Yes Renegotiation Attempt with Same Value

Purpose: To verify that an iSCSI target negotiates the InitialR2T key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.10

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: The InitialR2T key is used to toggle the use of R2T for the first outgoing data burst between initiator and target. InitialR2T is defined as a Session-wide parameter. An attempt to renegotiate/redeclare parameters not specifically allowed MUST be detected by the initiator and target. If such an attempt is detected by the target, the target MUST respond with Login reject (initiator error).

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate InitialR2T=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, offer the key=value pair InitialR2T=Yes.

Observable Results:
- Verify the DUT recognizes the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.5.4 InitialR2T=Yes Renegotiation Attempt with InitialR2T=No

Purpose: To verify that an iSCSI target negotiates the InitialR2T key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.10

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: February 16, 2009

Discussion: The InitialR2T key is used to toggle the use of R2T for the first outgoing data burst between initiator and target. InitialR2T is defined as a Session-wide parameter. An attempt to renegotiate/redeclare parameters not specifically allowed MUST be detected by the initiator and target. If a renegotiation is detected by a Target, the Target must answer with a Login Reject.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate InitialR2T=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Allow the DUT to open a new connection within the "Normal" session.
- On the second connection, offer the key=value pair InitialR2T=No.

Observable Results:
- Verify the DUT recognizes the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire
InterOperability Laboratory

Test #2.5.5 InitialR2T=No Renegotiation Attempt with Same Value

Purpose: To verify that an iSCSI target negotiates the InitialR2T key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.10

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: February 16, 2009

Discussion: The InitialR2T key is used to toggle the use of R2T for the first outgoing data burst between initiator and target. InitialR2T is defined as a Session-wide parameter. An attempt to renegotiate/redeclare parameters not specifically allowed MUST be detected by the initiator and target. If a renegotiation is detected by a Target, the Target must answer with a Login Reject.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate InitialR2T=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, offer the key=value pair InitialR2T=No.

Observable Results:
- Verify the DUT recognizes the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable. If the DUT does not support InitialR2T=No this item is not testable.
Test #2.5.6 InitialR2T=No Renegotiation Attempt with InitialR2T=Yes

Purpose: To verify that an iSCSI target negotiates the InitialR2T key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.10

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: February 16, 2009

Discussion: The InitialR2T key is used to toggle the use of R2T for the first outgoing data burst between initiator and target. InitialR2T is defined as a Session-wide parameter. An attempt to renegotiate/redeclare parameters not specifically allowed MUST be detected by the initiator and target. If a renegotiation is detected by a Target, the Target must answer with a Login Reject.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate InitialR2T=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, offer the key=value pair InitialR2T=Yes.

Observable Results:
- Verify the DUT recognizes the renegotiation and drops the connection.

Possible Problems: If the device does not support more than 1 connection this item is not testable. If the DUT does not support InitialR2T=No this item is not testable.
Test #2.6.1 ImmediateData=Yes Across Two Connections

Purpose: To verify that an iSCSI target negotiates the ImmediateData key properly when using multiple connections.

Reference: 12.11

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: The ImmediateData key determines if the target will accept ImmediateData in non-Data PDUs. ImmediateData is defined as a Session Wide parameter. If ImmediateData is set to Yes and InitialR2T is set to Yes (default), then only immediate data are accepted in the first burst. If ImmediateData is set to Yes and InitialR2T is set to No, then the initiator MAY send unsolicited immediate data and/or one unsolicited burst of Data-Out PDUs.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
• Open a session of type "Normal" from the Testing Station to the DUT.
• On the leading login of the session, negotiate ImmediateData=Yes.
• Complete the Login Phase and proceed to Full Feature Phase operation.
• Open a new connection within the "Normal" session.
• On the second connection complete the Login Phase, and proceed to Full Feature Phase.
• On each connection, perform a WRITE command.

Observable Results:
• Verify the DUT does not offer the ImmediateData key on the second connection.
• Verify that on each connection the DUT accepts ImmediateData with the WRITE command.

Possible Problems: If the device does not support more than 1 connection this item is not testable. If the DUT does not support ImmediateData=Yes the item is not testable.
Test #2.6.2 ImmediateData=No Across Two Connections

**Purpose:** To verify that an iSCSI target negotiates the ImmediateData key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 12.11

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 24, 2009

**Discussion:** The ImmediateData key determines if the target will accept ImmediateData in non-Data PDUs. ImmediateData is defined as a Session Wide parameter. If ImmediateData is set to No and InitialR2T is set to Yes, then the initiator MUST NOT send unsolicited data and the target MUST reject unsolicited data with the corresponding response code.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate ImmediateData=No, InitialR2T=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection complete the Login Phase, and proceed to Full Feature Phase.
- On each connection, perform a WRITE command with ImmediateData.

**Observable Results:**
- Verify the DUT does not offer the ImmediateData key on the second connection.
- Verify that on each connection the DUT rejects the unsolicited ImmediateData with the WRITE command.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.6.3 ImmediateData=Yes Renegotiation Attempt with Same Value

**Purpose:** To verify that an iSCSI target negotiates the ImmediateData key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 5.3, 12.11

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 27, 2007

**Discussion:** The ImmediateData key determines if the target will accept ImmediateData in non-Data PDUs. ImmediateData is defined as a Session Wide parameter. Targets are required to detect renegotiations, and send a Login Reject when they are detected.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate ImmediateData=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should offer the key=value ImmediateData=Yes.

**Observable Results:**
- Verify the DUT recognizes the renegotiation and transmits a Login Reject.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable. If the DUT does not support ImmediateData=Yes this item is not testable.
Test #2.6.4 ImmediateData=Yes Renegotiation Attempt with ImmediateData=No

**Purpose:** To verify that an iSCSI target negotiates the ImmediateData key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 5.3, 12.11

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 27, 2007

**Discussion:** The ImmediateData key determines if the target will accept ImmediateData in non-Data PDUs. ImmediateData is defined as a Session Wide parameter. Targets are required to detect renegotiations, and send a Login Reject when they are detected.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate ImmediateData=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should offer the key=value ImmediateData=No.

**Observable Results:**
- Verify the DUT recognizes the renegotiation and transmits a Login Reject.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable. If the DUT does not support ImmediateData=Yes this item is not testable.
Test #2.6.5 ImmediateData=No Renegotiation Attempt with Same Value

Purpose: To verify that an iSCSI target negotiates the ImmediateData key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.11

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 27, 2007

Discussion: The ImmediateData key determines if the target will accept ImmediateData in non-Data PDUs. ImmediateData is defined as a Session Wide parameter. Targets are required to detect renegotiations, and send a Login Reject when they are detected.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate ImmediateData=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should offer the key=value ImmediateData=No.

Observable Results:
- Verify the DUT recognizes the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire  
InterOperability Laboratory

Test #2.6.6 ImmediateData=No Renegotiation Attempt with ImmediateData=Yes

Purpose: To verify that an iSCSI target negotiates the ImmediateData key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.11

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 27, 2007

Discussion: The ImmediateData key determines if the target will accept ImmediateData in non-Data PDUs. ImmediateData is defined as a Session Wide parameter. Targets are required to detect renegotiations, and send a Login Reject when they are detected.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate ImmediateData=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should offer the key=value ImmediateData=Yes.

Observable Results:
- Verify the DUT recognizes the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.7.1 MaxRecvDataSegmentLength Declared on First Connection Only

Purpose: To verify that an iSCSI target uses the value declared by the initiator for MaxRecvDataSegmentLength.

Reference: iSCSI Standard Clause 12.12

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 29, 2007

Discussion: The MaxRecvDataSegmentLength key declares the maximum data segment length in bytes it can receive in an iSCSI PDU. MaxRecvDataSegmentLength is a connection-only parameter.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, declare MaxRecvDataSegmentLength=512.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should not declare a value for MaxRecvDataSegmentLength.
- On the second connection, complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection, perform a READ operation.

Observable Results:
- Verify that on the first connection the DUT uses the declared value of 512 for MaxRecvDataSegmentLength in its Data-In PDUs.
- On the second connection the DUT is expected but not required to use the default value of MaxRecvDataSegmentLength when sizing its PDUs. The DUT should not use the value of 512 for MaxRecvDataSegmentLength for its Data-In PDUs on the second connection.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.7.2 MaxRecvDataSegmentLength Declared on Second Connection Only

Purpose: To verify that an iSCSI target uses the value declared by the initiator for MaxRecvDataSegmentLength.

Reference: iSCSI Standard Clause 12.12

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: The MaxRecvDataSegmentLength key declares the maximum data segment length in bytes it can receive in an iSCSI PDU. MaxRecvDataSegmentLength is a connection-only parameter.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, do not declare a value for MaxRecvDataSegmentLength.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should declare a value for MaxRecvDataSegmentLength=512.
- On the second connection, complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection, the Testing Station should attempt to perform a READ operation.

Observable Results:
- Verify that on the second connection the DUT uses the declared value of 512 for MaxRecvDataSegmentLength in its Data-In PDUs.
- On the first connection, the DUT is expected but not required to use the default value of MaxRecvDataSegmentLength when sizing its PDUs. The DUT should not use the value of 512 for MaxRecvDataSegmentLength for its Data-In PDUs on the first connection.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.7.3 MaxRecvDataSegmentLength Declared with Larger Value on Second Connection

Purpose: To verify that an iSCSI initiator uses the value declared by the initiator for MaxRecvDataSegmentLength.

Reference: iSCSI Standard Clause 12.12

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: The MaxRecvDataSegmentLength key declares the maximum data segment length in bytes it can receive in an iSCSI PDU. MaxRecvDataSegmentLength is a connection-only parameter.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, declare MaxRecvDataSegmentLength=512.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should declare MaxRecvDataSegmentLength=1024.
- On the second connection, complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection, the Testing Station should attempt to perform a READ operation.

Observable Results:
- Verify that on the first connection the DUT uses the declared value of 512 for MaxRecvDataSegmentLength for its Data-In PDUs.
- On the second connection, the DUT should use the value of 1024 for MaxRecvDataSegmentLength for its Data-In PDUs. The DUT is not required to use the largest possible value for DataSegmentLength, and may therefore transmit Data PDUs with less than 1024 bytes.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.7.4 MaxRecvDataSegmentLength Declared with Smaller Value on Second Connection

Purpose: To verify that an iSCSI target uses the value declared by the initiator for MaxRecvDataSegmentLength.

Reference: iSCSI Standard Clause 12.12

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: The MaxRecvDataSegmentLength key declares the maximum data segment length in bytes it can receive in an iSCSI PDU. MaxRecvDataSegmentLength is a connection-only parameter.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, declare MaxRecvDataSegmentLength=1024.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should declare MaxRecvDataSegmentLength=512.
- On the second connection, complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection, the DUT should attempt to perform a READ operation.

Observable Results:
- Verify that on the second connection the DUT uses the declared value of 512 for MaxRecvDataSegmentLength for its Data-In PDUs.
- On the first connection, the DUT should use the value of 1024 for MaxRecvDataSegmentLength for its Data-In PDUs. The DUT is not required to use the largest possible value for DataSegmentLength, and may therefore transmit Data PDUs with less than 1024 bytes.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.8.1 MaxBurstLength Key

**Purpose:** To verify that an iSCSI target properly negotiates and uses MaxBurstLength when using multiple connections.

**Reference:** iSCSI Standard Clause 12.13

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 2, 2007

**Discussion:** The MaxBurstLength key is used to negotiate the maximum SCSI data payload in bytes in a Data-In or a solicited Data-Out iSCSI sequence. MaxBurstLength is a Leading Only parameter; it can only be negotiated in the leading connection of a session.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate MaxBurstLength=512.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection, the Testing Station should attempt to perform a READ operation.

**Observable Results:**
- Verify that during the Login Phase of the second connection the DUT does not attempt to renegotiate the MaxBurstLength key.
- Verify that on each connection the DUT uses the value of 512 for MaxBurstLength, not transmitting more than 512 bytes in a sequence before transmitting a Data-In with the F-bit set.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.8.2 MaxBurstLength Renegotiation Attempt

Purpose: To verify that an iSCSI target properly negotiates and uses MaxBurstLength when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.13

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 2, 2007

Discussion: The MaxBurstLength key is used to negotiate the maximum SCSI data payload in bytes in a Data-In or a solicited Data-Out iSCSI sequence. MaxBurstLength is a Leading Only parameter, it can only be negotiated in the leading connection of a session. An iSCSI target must detect renegotiations, and if one occurs, send a Login Reject.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate MaxBurstLength=512.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should attempt to negotiate MaxBurstLength=512.

Observable Results:
- Verify that the DUT recognizes the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire
InterOperability Laboratory

Test #2.9.1 FirstBurstLength Key

Purpose: To verify that an iSCSI target properly negotiates and uses FirstBurstLength when using multiple connections.

Reference: iSCSI Standard Clause 12.14

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 2, 2007

Discussion: The FirstBurstLength key is used to negotiate the maximum amount of unsolicited data in bytes an initiator may send to a target during the execution of a SCSI command. FirstBurstLength is a Leading Only parameter; it can only be negotiated in the leading connection of a session.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate FirstBurstLength=512, ImmediateData=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection, the DUT should attempt to perform a WRITE operation with more than 512 bytes of ImmediateData attached.

Observable Results:
- Verify that during the Login Phase of the second connection the DUT does not attempt to renegotiate the FirstBurstLength key.
- Verify that on each connection the DUT sees the error of sending too much unsolicited data.

Possible Problems: If the device does not support more than 1 connection this item is not testable. The DUT may not support ImmediateData=Yes.
Test #2.9.2 FirstBurstLength Renegotiation Attempt

Purpose: To verify that an iSCSI target properly negotiates and uses FirstBurstLength when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.14

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 2, 2007

Discussion: The FirstBurstLength key is used to negotiate the maximum amount of unsolicited data in bytes an initiator may send to a target during the execution of a SCSI command. FirstBurstLength is a Leading Only parameter; it can only be negotiated in the leading connection of a session. An iSCSI target must detect renegotiations, and if one occurs, send a Login Reject.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" between the Testing Station and the DUT.
- On the leading login of the session, negotiate FirstBurstLength=512.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should attempt to negotiate FirstBurstLength=512.

Observable Results:
- Verify that the DUT recognizes the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.10.1 DefaultTime2Wait Renegotiation Check

Purpose: To verify that an iSCSI target properly negotiates and uses DefaultTime2Wait when using multiple connections.

Reference: iSCSI Standard Clause 12.15

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 2, 2007

Discussion: The DefaultTime2Wait key lets the initiator and target negotiate the minimum time, in seconds, to wait before attempting an explicit/implicit logout or an active task reassignment after an unexpected connection termination or a connection reset.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal".
- On the leading login of the session, negotiate DefaultTime2Wait=4.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.

Observable Results:
- Verify that the DUT does not attempt to renegotiate DefaultTime2Wait on the second connection.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.10.2 DefaultTime2Wait Renegotiation Attempt

**Purpose:** To verify that an iSCSI target properly negotiates and uses DefaultTime2Wait when using multiple connections.

**Reference:** iSCSI Standard Clause 5.3, 12.15

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 2, 2007

**Discussion:** The DefaultTime2Wait key lets the initiator and target negotiate the minimum time, in seconds, to wait before attempting an explicit/implicit logout or an active task reassignment after an unexpected connection termination or a connection reset. If a target detects a renegotiation it must send a Login Reject. Renegotiations must be detected if they occur.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DefaultTime2Wait=4.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection of the session, negotiate DefaultTime2Wait=8.

**Observable Results:**
- Verify that the DUT recognizes the renegotiation and transmits a Login Reject.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.11.1 DefaultTime2Retain Renegotiation Check

Purpose: To verify that an iSCSI target properly negotiates and uses DefaultTime2Retain when using multiple connections.

Reference: 5.3, 12.16

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 2, 2007

Discussion: The DefaultTime2Retain key lets the initiator and target negotiate the maximum time, in seconds after an initial wait (Time2Wait), before which an active task reassignment is still possible after an unexpected connection termination or a connection reset.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:

- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DefaultTime2Retain=10.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.

Observable Results:

- Verify that the DUT does not attempt to renegotiate DefaultTime2Retain in the second connection.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.11.2 DefaultTime2Retain Renegotiation Attempt

Purpose: To verify that an iSCSI target properly negotiates and uses DefaultTime2Retain when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.16

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 2, 2007

Discussion: The DefaultTime2Retain key lets the initiator and target negotiate the maximum time, in seconds after an initial wait (Time2Wait), before which an active task reassignment is still possible after an unexpected connection termination or a connection reset.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DefaultTime2Retain=10.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection of the session, negotiate DefaultTime2Retain=5.

Observable Results:
- Verify that the DUT recognizes the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.12.1 TaskReporting Renegotiation Check

**Purpose:** To verify that a device properly negotiates values for TaskReporting key on every connection.

**Reference:** iSCSI Corrections and Clarifications 9.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 7, 2009

**Discussion:** The TaskReporting key is used to negotiate the task completion reporting semantics from the SCSI target. Valid values for the TaskReporting key are RFC3720, ResponseFence, and FastAbort. TaskReporting is defined as a Session Wide parameter. Neither the initiator nor the target should attempt to declare or negotiate a parameter more than once during login except for responses to specific keys that explicitly allow repeated key declarations (e.g., TargetAddress).

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- In the leading login of a session, if not offered by the DUT, offer key=value pairs TaskReporting=ResponseFence,RFC3720 and MaxConnections=2.
- Open another connection to the DUT.

**Observable Results:**
- Verify that the DUT does not attempt to renegotiate the TaskReporting key in the second connection.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable. The TaskReporting key is new in RFC 5048. A device that is not compliant with RFC 5048 will not support TaskReporting.
**Test #2.12.2 TaskReporting Renegotiation Attempt**

**Purpose:** To verify that a device properly negotiates values for TaskReporting key on every connection.

**Reference:** iSCSI Corrections and Clarifications 9.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 7, 2009

**Discussion:** The TaskReporting key is used to negotiate the task completion reporting semantics from the SCSI target. Valid values for the TaskReporting key are RFC3720, ResponseFence, and FastAbort. TaskReporting is defined as a Session Wide parameter. Neither the initiator nor the target should attempt to declare or negotiate a parameter more than once during login except for responses to specific keys that explicitly allow repeated key declarations (e.g., TargetAddress). An attempt to renegotiate/redeclare parameters not specifically allowed MUST be detected by the initiator and target. If such an attempt is detected by the target, the target MUST respond with Login reject (initiator error).

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- In the leading login of a session, if not offered by the DUT, offer key=value pairs TaskReporting=ResponseFence,RFC3720 and MaxConnections=2.
- Open another connection to the DUT.
- During login phase of the second connection of the session, attempt to negotiate TaskReporting=FastAbort.

**Observable Results:**
- Verify that the DUT sends a Login Reject (Initiator Error).

**Possible Problems:** If the device does not support more than 1 connection this item is not testable. The TaskReporting key is new in RFC 5048. A device that is not compliant with RFC 5048 will not support TaskReporting.
Test #2.13.1 MaxOutstandingR2T Renegotiation Check

**Purpose:** To verify that an iSCSI target properly negotiates and uses MaxOutstandingR2T when using multiple connections.

**Reference:** iSCSI Standard Clause 12.17

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 24, 2009

**Discussion:** The MaxOutstandingR2T key lets the Initiator and target negotiate the maximum number of outstanding R2Ts per task, excluding any implied initial R2T that might be part of that task. An R2T is considered outstanding until the last data PDU (with the F bit set to 1) is transferred, or a sequence reception timeout (Section 6.1.4.1 Recovery Within-command) is encountered for that data sequence. MaxOutstandingR2T is defined as a Session-Wide parameter.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate MaxOutstandingR2T=4.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.

**Observable Results:**
- Verify that the DUT does not attempt to renegotiate MaxOutstandingR2T in the second connection.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.13.2 MaxOutstandingR2T Renegotiation Attempt

Purpose: To verify that an iSCSI target properly negotiates and uses MaxOutstandingR2T when using multiple connections.

Reference: 5.3, 12.17

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: The MaxOutstandingR2T key lets the Initiator and target negotiate the maximum number of outstanding R2Ts per task, excluding any implied initial R2T that might be part of that task. An R2T is considered outstanding until the last data PDU (with the F bit set to 1) is transferred, or a sequence reception timeout (Section 6.1.4.1 Recovery Within-command) is encountered for that data sequence. MaxOutstandingR2T is defined as a Session-Wide parameter. Targets are required to detect renegotiations, and send a Login Reject when renegotiations occur.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading connection of the session, negotiate MaxOutstandingR2T=4.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection of the session, negotiate MaxOutstandingR2T=4.

Observable Results:
- Verify that the DUT detects the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.13.3 MaxOutstandingR2T Key

Purpose: To verify that an iSCSI target properly negotiates and uses MaxOutstandingR2T when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.17

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: The MaxOutstandingR2T key lets the initiator and target negotiate the maximum number of outstanding R2Ts per task, excluding any implied initial R2T that might be part of that task. An R2T is considered outstanding until the last data PDU (with the F bit set to 1) is transferred, or a sequence reception timeout (Section 6.1.4.1 Recovery Within-command) is encountered for that data sequence. MaxOutstandingR2T is defined as a Session-Wide parameter. Targets are required to detect renegotiations, and send a Login Reject when renegotiations occur.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate MaxOutstandingR2T=4, MaxBurstLength=1024, FirstBurstLength=1024.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection of the session, complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection, the Testing Station should start a WRITE operation to transfer at least 4096 bytes.
- When the WRITE command is received, the DUT should transmit up to 4 R2Ts on the same connection as the command was received for 1024 bytes each.
- The DUT should transmit more R2Ts on the same connection each time a Data-Out from a WRITE command is received, until R2T's for all 4096 bytes have been transmitted.

Observable Results:
- Verify that the DUT completes each WRITE operation successfully.

Possible Problems: If the device does not support more than 1 connection this item is not testable. If the device does not support more than 1 outstanding R2T this item is not testable. The DUT is not required to send outstanding R2T's. If the DUT does not send outstanding R2T's this item is not testable.
Test #2.14.1 DataPDUInOrder=Yes Across Two Connections

Purpose: To verify that an iSCSI target negotiates the DataPDUInOrder key properly when using multiple connections.

Reference: iSCSI Standard Clause 12.18

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: DataPDUInOrder=No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. DataPDUInOrder=Yes is used to indicate that data PDUs within sequences have to be at continuously increasing addresses and overlays are forbidden. The DataPDUInOrder key is a session-wide key and is only allowed to be used in the leading connection of a session.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataPDUInOrder=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, complete the Login Phase, and proceed to Full Feature Phase.
- On each connection, the Testing Station should to perform a READ command.

Observable Results:
- Verify the DUT does not offer the DataPDUInOrder key on the second connection.
- Verify that on each connection the DUT transmits Data-Out PDUs with increasing addresses and without overlays.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.14.2.1 DataPDUInOrder=No Across Two Connections With InitialR2T=Yes

Purpose: To verify that an iSCSI target negotiates the DataPDUInOrder key properly when using multiple connections.

Reference: iSCSI Standard Clause 12.18

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: DataPDUInOrder=No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. DataPDUInOrder=Yes is used to indicate that data PDUs within sequences have to be at continuously increasing addresses and overlays are forbidden. The DataPDUInOrder key is a session-wide key and is only allowed to be used in the leading connection of a session.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataPDUInOrder=No, InitialR2T=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, complete the Login Phase, and proceed to Full Feature Phase.
- On each connection, the Testing Station should perform a WRITE command.
- The Testing Station should transmit Data-Out PDUs for the WRITE once R2T is received. The Data-Out PDU's should not have continuously increasing addresses, but rather be out of order.

Observable Results:
- Verify the DUT does not offer the DataPDUInOrder key on the second connection.
- Verify that on each connection the DUT accepts the Data-Out PDUs. The WRITE command should complete with GOOD status.

Possible Problems: If the device does not support more than 1 connection this item is not testable. If the DUT does not support DataPDUInOrder=No this item is not testable.
Test #2.14.2.2 DataPDUInOrder=No Across Two Connections with InitialR2T=No

**Purpose:** To verify that an iSCSI target negotiates the DataPDUInOrder key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 12.18

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 24, 2009

**Discussion:** DataPDUInOrder=No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. DataPDUInOrder=Yes is used to indicate that data PDUs within sequences have to be at continuously increasing addresses and overlays are forbidden. The DataPDUInOrder key is a session-wide key and is only allowed to be used in the leading connection of a session.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataPDUInOrder=No, InitialR2T=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, complete the Login Phase, and proceed to Full Feature Phase.
- On each connection, the Testing Station should perform a WRITE command.
- The Testing Station should transmit Data-Out PDUs for the WRITE immediately. The Data-Out PDU's should not have continuously increasing addresses, but rather be out of order.

**Observable Results:**
- Verify the DUT does not offer the DataPDUInOrder key on the second connection.
- Verify that on each connection the DUT accepts the Data-Out PDUs. The WRITE command should complete with GOOD status.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable. If the DUT does not support DataPDUInOrder=No and InitialR2T=No this item is not testable.
Test #2.14.3 DataPDUInOrder=Yes Renegotiation Attempt with Same Value

**Purpose:** To verify that an iSCSI target negotiates the DataPDUInOrder key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 5.3, 12.18

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 24, 2009

**Discussion:** DataPDUInOrder=No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. DataPDUInOrder=Yes is used to indicate that data PDUs within sequences have to be at continuously increasing addresses and overlays are forbidden. The DataPDUInOrder key is a session-wide key and is only allowed to be used in the leading connection of a session. Targets are required to detect the renegotiation of keys. If a renegotiation is detected, the Target must send a Login Reject.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataPDUInOrder=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, offer DataPDUInOrder=Yes.

**Observable Results:**
- Verify the DUT detects the renegotiation and transmits a Login Reject.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.14.4 DataPDUInOrder=Yes Renegotiation Attempt with DataPDUInOrder=No

Purpose: To verify that an iSCSI target negotiates the DataPDUInOrder key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.18

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: DataPDUInOrder=No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. DataPDUInOrder=Yes is used to indicate that data PDUs within sequences have to be at continuously increasing addresses and overlays are forbidden. The DataPDUInOrder key is a session-wide key and is only allowed to be used in the leading connection of a session. Targets are required to detect the renegotiation of keys. If a renegotiation is detected, the Target must send a Login Reject.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading connection of the session, negotiate DataPDUInOrder=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, offer DataPDUInOrder=No.

Observable Results:
- Verify the DUT detects the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire  
InterOperability Laboratory  

Test #2.14.5 DataPDUInOrder=No Renegotiation Attempt with Same Value

**Purpose:** To verify that an iSCSI target negotiates the DataPDUInOrder key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 5.3, 12.18

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** June 24, 2009

**Discussion:** DataPDUInOrder=No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. DataPDUInOrder=Yes is used to indicate that data PDUs within sequences have to be at continuously increasing addresses and overlays are forbidden. The DataPDUInOrder key is a session-wide key and is only allowed to be used in the leading connection of a session. Targets are required to detect the renegotiation of keys. If a renegotiation is detected, the Target must send a Login Reject.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataPDUInOrder=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, offer DataPDUInOrder=No.

**Observable Results:**
- Verify the DUT detects the renegotiation and transmits Login Reject.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire
InterOperability Laboratory

Test #2.14.6 DataPDUInOrder=No Renegotiation Attempt with DataPDUInOrder=Yes

Purpose: To verify that an iSCSI target negotiates the DataPDUInOrder key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 12.18

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: June 24, 2009

Discussion: DataPDUInOrder=No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. DataPDUInOrder=Yes is used to indicate that data PDUs within sequences have to be at continuously increasing addresses and overlays are forbidden. The DataPDUInOrder key is a session-wide key and is only allowed to be used in the leading connection of a session. Targets are required to detect the renegotiation of keys. If a renegotiation is detected, the Target must send a Login Reject.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataPDUInOrder=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, offer DataPDUInOrder=Yes.

Observable Results:
- Verify the DUT detects the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.15.1 READ Command when DataSequenceInOrder=Yes Across Two Connections

**Purpose:** To verify that an iSCSI target negotiates the DataSequenceInOrder key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 10.7.6, 12.19

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 5, 2007

**Discussion:** The ordering between sequences is determined by DataSequenceInOrder. When set to Yes, it means that sequences have to be in increasing Buffer Offset order and overlays are forbidden. A Data Sequence is a sequence of Data-In or Data-Out PDUs that end with a Data-In or Data-Out PDU with the F bit set to one. A Data-Out sequence is sent either unsolicited or in response to an R2T. Sequences cover an offset-range. If DataSequenceInOrder is set to No, Data PDU sequences may be transferred in any order. If DataSequenceInOrder is set to Yes, Data Sequences MUST be transferred using continuously non-decreasing sequence offsets (R2T buffer offset for writes, or the smallest SCSI Data-In buffer offset within a read data sequence). DataSequenceInOrder is a session wide parameter only to be used in the leading connection of a session.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataSequenceInOrder=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection complete the Login Phase, and proceed to Full Feature Phase.
- On each connection the Testing Station should perform a READ command.

**Observable Results:**
- Verify the DUT does not offer the DataSequenceInOrder key on the second connection.
- Verify that on each connection the DUT transmits Data-In Data Sequences using continuously non-decreasing sequence offsets.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable. The DUT is not required to transmit Data Sequences out of order, this is optional.
Test #2.15.2 WRITE Command when DataSequenceInOrder=Yes Across Two Connections

**Purpose:** To verify that an iSCSI target negotiates the DataSequenceInOrder key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 10.7.6, 12.19

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 5, 2007

**Discussion:** The ordering between sequences is determined by DataSequenceInOrder. When set to Yes, it means that sequences have to be in increasing Buffer Offset order and overlays are forbidden. A Data Sequence is a sequence of Data-In or Data-Out PDUs that end with a Data-In or Data-Out PDU with the F bit set to one. A Data-out sequence is sent either unsolicited or in response to an R2T. Sequences cover an offset-range. If DataSequenceInOrder is set to No, Data PDU sequences may be transferred in any order. If DataSequenceInOrder is set to Yes, Data Sequences MUST be transferred using continuously non-decreasing sequence offsets (R2T buffer offset for writes, or the smallest SCSI Data-In buffer offset within a read data sequence). DataSequenceInOrder is a session wide parameter only to be used in the leading connection of a session.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataSequenceInOrder=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, complete the Login Phase, and proceed to Full Feature Phase.
- On each connection, allow the Testing Station should perform a WRITE command.

**Observable Results:**
- Verify the DUT does not offer the DataSequenceInOrder key on the second connection.
- Verify that on each connection the DUT transmits R2Ts with continuously increasing buffer offset.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Purpose: To verify that an iSCSI target negotiates the DataSequenceInOrder key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 10.7.6, 12.19

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 5, 2007

Discussion: The ordering between sequences is determined by DataSequenceInOrder. When set to Yes, it means that sequences have to be in increasing Buffer Offset order and overlays are forbidden. A Data Sequence is a sequence of Data-In or Data-Out PDUs that end with a Data-In or Data-Out PDU with the F bit set to one. A Data-out sequence is sent either unsolicited or in response to an R2T. Sequences cover an offset-range. If DataSequenceInOrder is set to No, Data PDU sequences may be transferred in any order. If DataSequenceInOrder is set to Yes, Data Sequences MUST be transferred using continuously non-decreasing sequence offsets (R2T buffer offset for writes, or the smallest SCSI Data-In buffer offset within a read data sequence). DataSequenceInOrder is a session wide parameter only to be used in the leading connection of a session. Targets are required to detect the renegotiation of keys. If a renegotiation is detected, the Target must send a Login Reject.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataSequenceInOrder=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should offer DataSequenceInOrder=Yes.

Observable Results:
- Verify the DUT detects the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Purpose: To verify that an iSCSI target negotiates the DataSequenceInOrder key properly when using multiple connections.

Reference: iSCSI Standard Clause 5.3, 10.7.6, 12.19

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 5, 2007

Discussion: The ordering between sequences is determined by DataSequenceInOrder. When set to Yes, it means that sequences have to be in increasing Buffer Offset order and overlays are forbidden. A Data Sequence is a sequence of Data-In or Data-Out PDUs that end with a Data-In or Data-Out PDU with the F bit set to one. A Data-out sequence is sent either unsolicited or in response to an R2T. Sequences cover an offset-range. If DataSequenceInOrder is set to No, Data PDU sequences may be transferred in any order. If DataSequenceInOrder is set to Yes, Data Sequences MUST be transferred using continuously non-decreasing sequence offsets (R2T buffer offset for writes, or the smallest SCSI Data-In buffer offset within a read data sequence). DataSequenceInOrder is a session wide parameter only to be used in the leading connection of a session. Targets are required to detect the renegotiation of keys. If a renegotiation is detected, the Target must send a Login Reject.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataSequenceInOrder=Yes.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection the Testing Station should offer DataSequenceInOrder=No.

Observable Results:
- Verify the DUT detects the renegotiation and transmits a Login Reject.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire
InterOperability Laboratory

Test #2.15.5 DataSequenceInOrder=No Renegotiation Attempt with Same Value

**Purpose:** To verify that an iSCSI target negotiates the DataSequenceInOrder key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 5.3, 10.7.6, 12.19

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 5, 2007

**Discussion:** The ordering between sequences is determined by DataSequenceInOrder. When set to Yes, it means that sequences have to be in increasing Buffer Offset order and overlays are forbidden. A Data Sequence is a sequence of Data-In or Data-Out PDUs that end with a Data-In or Data-Out PDU with the F bit set to one. A Data-out sequence is sent either unsolicited or in response to an R2T. Sequences cover an offset-range. If DataSequenceInOrder is set to No, Data PDU sequences may be transferred in any order. If DataSequenceInOrder is set to Yes, Data Sequences MUST be transferred using continuously non-decreasing sequence offsets (R2T buffer offset for writes, or the smallest SCSI Data-In buffer offset within a read data sequence). DataSequenceInOrder is a session wide parameter only to be used in the leading connection of a session. Targets are required to detect the renegotiation of keys. If a renegotiation is detected, the Target must send a Login Reject.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station.
- On the leading login of the session, negotiate DataSequenceInOrder=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should offer DataSequenceInOrder=No.

**Observable Results:**
- Verify the DUT detects the renegotiation and transmits Login Reject.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire
InterOperability Laboratory

Test #2.15.6 DataSequenceInOrder=No Renegotiation Attempt with DataSequenceInOrder=Yes

**Purpose:** To verify that an iSCSI target negotiates the DataSequenceInOrder key properly when using multiple connections.

**Reference:** iSCSI Standard Clause 5.3, 10.7.6, 12.19

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 5, 2007

**Discussion:** The ordering between sequences is determined by DataSequenceInOrder. When set to Yes, it means that sequences have to be in increasing Buffer Offset order and overlays are forbidden. A Data Sequence is a sequence of Data-In or Data-Out PDUs that end with a Data-In or Data-Out PDU with the F bit set to one. A Data-out sequence is sent either unsolicited or in response to an R2T. Sequences cover an offset-range. If DataSequenceInOrder is set to No, Data PDU sequences may be transferred in any order. If DataSequenceInOrder is set to Yes, Data Sequences MUST be transferred using continuously non-decreasing sequence offsets (R2T buffer offset for writes, or the smallest SCSI Data-In buffer offset within a read data sequence). DataSequenceInOrder is a session wide parameter only to be used in the leading connection of a session. Targets are required to detect the renegotiation of keys. If a renegotiation is detected, the Target must send a Login Reject.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DataSequenceInOrder=No.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- On the second connection, the Testing Station should offer DataSequenceInOrder=Yes.

**Observable Results:**
- Verify the DUT detects the renegotiation and transmits Login Reject.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.16.1 HeaderDigest and DataDigest Keys Across Two Connections

Purpose: To verify that a device properly negotiates values for Header and Data Digests on every connection.

Reference: iSCSI Standard Clause 12.1

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 5, 2007

Discussion: Header and Data Digests are connection-only parameters. Thus their values must be negotiated for every open connection.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- In the leading login of a session, if not offered by the DUT, offer key=value pairs HeaderDigest=CRC32C,None and DataDigest=CRC32C,None and MaxConnections=2.
- Open another connection to the DUT.
- During login phase of the second connection of the session, if not offered by the DUT, offer key=value pairs HeaderDigest=CRC32C,None, and DataDigest=CRC32C,None.
- Proceed to Full Feature Phase on each connection. Generate SCSI traffic on each connection.

Observable Results:
- Verify that the DUT accepts negotiation of connection-only parameters on each connection, and does not interpret this as re-negotiation.
- Verify that on each connection the DUT is including Header and Data Digests in the PDUs it transmits and that these Digests are calculated correctly.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #2.16.2 HeaderDigest and DataDigest Keys Across Three Connections

Purpose: To verify that a device properly negotiates values for Header and Data Digests on every connection.

Reference: iSCSI Standard Clause 12.1

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 5, 2007

Discussion: Header and Data Digests are connection-only parameters. Thus their values must be negotiated for every open connection.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- In the leading login of a session, if not offered by the DUT, offer key=value pairs HeaderDigest=CRC32C,None and DataDigest=CRC32C,None and MaxConnections=3.
- Open another connection to the DUT.
- During login phase of the second connection of the session, if not offered by the DUT, offer key=value pairs HeaderDigest=CRC32C,None, and DataDigest=CRC32C,None.
- Proceed to Full Feature Phase on each connection. Generate SCSI traffic on each connection.
- If possible allow the DUT to open third connection to the Testing Stations.
- During login phase of the third connection of the session, if not offered by the DUT, offer key=value pairs HeaderDigest=CRC32C,None, and DataDigest=CRC32C,None.
- Proceed to Full Feature Phase on the third connection. Generate SCSI traffic on each connection.

Observable Results:
- Verify that the DUT accepts negotiation of connection-only parameters on each connection, and does not interpret this as re-negotiation.
- Verify that on each connection the DUT is including Header and Data Digests in the PDUs it transmits and that these Digests are calculated correctly.

Possible Problems: If the device does not support more than 2 connections this item is not testable.
Overview: This group of tests verifies the sequence number ordering and handling of multiconnection iSCSI sessions as defined in RFC 3720 and RFC 5048. Comments and questions regarding the implementation of these tests are welcome, and may be forwarded to Peter Scruton, UNH InterOperability Lab (pjs@iol.unh.edu).
The University of New Hampshire  
InterOperability Laboratory

Test #3.1.1 In Order CmdSN Wraparound Across Two Connections

Purpose: To verify that a device using multiple connections is able to complete SCSI commands on the open connections and properly sets the sequence numbers of each transmitted PDU.

Reference: iSCSI Standard Clause 3.2.2.1

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: February 16, 2009

Discussion: For the numbering mechanism, the initiator and target maintain the following three variables for each session:
CmdSN - the current command Sequence Number, advanced by 1 on each command shipped except for commands marked for immediate delivery. CmdSN always contains the number to be assigned to the next Command PDU. ExpCmdSN - the next expected command by the target. The target acknowledges all commands up to, but not including, this number. The initiator treats all commands with CmdSN less than ExpCmdSN as acknowledged. The target iSCSI layer sets the ExpCmdSN to the largest non-immediate CmdSN that it can deliver for execution plus 1 (no holes in the CmdSN sequence). MaxCmdSN - the maximum number to be shipped. The queuing capacity of the receiving iSCSI layer is MaxCmdSN - ExpCmdSN + 1. The initiator's ExpCmdSN and MaxCmdSN are derived from target-to-initiator PDU fields.

The command number is carried by the iSCSI PDU as CmdSN (Command Sequence Number). The numbering is session-wide. Outgoing iSCSI PDUs carry this number. The iSCSI initiator allocates CmdSNs with a 32-bit unsigned counter (modulo 2**32). Comparisons and arithmetic on CmdSN use Serial Number Arithmetic as defined in [RFC1982] where SERIAL_BITS = 32.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=2.
- The first Login Request PDU from the Testing Station should have a CmdSN value of no more than 5 less than the maximum value for CmdSN (4294967296).
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second connection of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On the first connection, transmit a READ command. Continue to perform READ Commands on the first connection until the maximum possible CmdSN is reached (4294967296).
- Transmit a READ Command with a CmdSN of 0 on the second connection.

Observable Results:
- Verify that SCSI traffic appears on each connection, and that each SCSI command completes successfully.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire
InterOperability Laboratory

Test #3.1.2 Out of Order CmdSN Wraparound Across Two Connections

**Purpose:** To verify that a device using multiple connections is able to complete SCSI commands on the open connections and properly sets the sequence numbers of each transmitted PDU.

**Reference:** iSCSI Standard Clause 3.2.2.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** February 16, 2009

**Discussion:** For the numbering mechanism, the initiator and target maintain the following three variables for each session:

- **CmdSN** - the current command Sequence Number, advanced by 1 on each command shipped except for commands marked for immediate delivery. CmdSN always contains the number to be assigned to the next Command PDU. ExpCmdSN - the next expected command by the target. The target acknowledges all commands up to, but not including, this number. The initiator treats all commands with CmdSN less than ExpCmdSN as acknowledged. The target iSCSI layer sets the ExpCmdSN to the largest non-immediate CmdSN that it can deliver for execution plus 1 (no holes in the CmdSN sequence). MaxCmdSN - the maximum number to be shipped. The queuing capacity of the receiving iSCSI layer is MaxCmdSN - ExpCmdSN + 1. The initiator's ExpCmdSN and MaxCmdSN are derived from target-to-initiator PDU fields.

The command number is carried by the iSCSI PDU as CmdSN (Command Sequence Number). The numbering is session-wide. Outgoing iSCSI PDUs carry this number. The iSCSI initiator allocates CmdSNs with a 32-bit unsigned counter (modulo 2**32). Comparisons and arithmetic on CmdSN use Serial Number Arithmetic as defined in [RFC1982] where SERIAL_BITS = 32.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=2.
- The first Login Request PDU from the Testing Station should have a CmdSN value of no more than 5 less than the maximum value for CmdSN (4294967296).
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second connection of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On the first connection, transmit a READ command. Continue to perform READ Commands on the first connection until 1 less than the maximum possible CmdSN is reached (4294967295).
- Transmit READ Commands with CmdSNs of 0 and 1 on the second connection.
- Transmit READ Commands with CmdSNs of 4294967296, 2, and 3 on the first connection.

**Observable Results:**
- Verify that SCSI traffic appears on each connection, and that each SCSI command completes successfully.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #3.1.3 In Order CmdSN Wraparound on One Connection

Purpose: To verify that a device using multiple connections is able to complete SCSI commands on the open connections and properly sets the sequence numbers of each transmitted PDU.

Reference: 3.2.2.1

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: February 16, 2009

Discussion: For the numbering mechanism, the initiator and target maintain the following three variables for each session: CmdSN - the current command Sequence Number, advanced by 1 on each command shipped except for commands marked for immediate delivery. CmdSN always contains the number to be assigned to the next Command PDU. ExpCmdSN - the next expected command by the target. The target acknowledges all commands up to, but not including, this number. The initiator treats all commands with CmdSN less than ExpCmdSN as acknowledged. The target iSCSI layer sets the ExpCmdSN to the largest non-immediate CmdSN that it can deliver for execution plus 1 (no holes in the CmdSN sequence). MaxCmdSN - the maximum number to be shipped. The queuing capacity of the receiving iSCSI layer is MaxCmdSN - ExpCmdSN + 1. The initiator's ExpCmdSN and MaxCmdSN are derived from target-to-initiator PDU fields.

The command number is carried by the iSCSI PDU as CmdSN (Command Sequence Number). The numbering is session-wide. Outgoing iSCSI PDUs carry this number. The iSCSI initiator allocates CmdSNs with a 32-bit unsigned counter (modulo 2**32). Comparisons and arithmetic on CmdSN use Serial Number Arithmetic as defined in [RFC1982] where SERIAL_BITS = 32.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=2.
- The first Login Request PDU from the Testing Station should have a CmdSN value of no more than 5 less than the maximum value for CmdSN (4294967296).
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second connection of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On the first connection, transmit a READ command. Continue to perform READ Commands on the first connection until the CmdSN wraps back to 0.
- Transmit READ Commands with CmdSNs of 1 and 2 on the second connection.

Observable Results:
- Verify that SCSI traffic appears on each connection, and that each SCSI command completes successfully.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #3.2.1: Duplicate CmdSN with Two Connections (First Command in Session)

**Purpose:** To verify that an iSCSI target ignores any non-immediate commands with a duplicate CmdSN inside of the range specified by MaxCmdSN to ExpCmdSN with multiple connections. In this test the duplicate command is the first command in a session.

**Reference:** iSCSI Standard Clause 3.2.2.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** January 11, 2008

**Discussion:** For non-immediate commands, the CmdSN field can take any value from ExpCmdSN to MaxCmdSN. The target MUST silently ignore any non-immediate command outside of the range or non-immediate duplicates within the range.

**Test Setup:** The DUT and Test Station pair should be able to make a TCP connection.

**Procedure:**
- Start two connections from the Testing Station to the iSCSI target being tested.
- During login negotiate the following parameters: ImmediateData=No, InitialR2T=Yes, MaxConnections=2.
- Complete the Login Phase and proceed to the Full Feature Phase.
- On the first connection: issue a series of two SCSI WRITE commands with a CmdSN of X+1, each with ExpectedDataTransferLength equal to the MaxRecvDataSegmentLength of the Testing Station. Where X is the ExpCmdSN after the Login Phase completes.
- On the second connection: issue a SCSI WRITE command with a CmdSN of X in such a way that the DUT receives this WRITE command after the first two WRITE commands. Where X is the ExpCmdSN after the Login Phase completes.
- After all WRITE commands have been transmitted, wait for an R2T for each command and respond to each R2T with Data-Out PDUs as necessary.

**Observable Results:**
- Verify that the target ignores the command with the duplicate CmdSN value.
- Verify that the target still properly handles the subsequently received WRITE commands.

**Possible Problems:** If the DUT does not support MaxConnections > 1 this item is not testable.
Test #3.2.2: Duplicate CmdSN with Two Connections (Not First Command in Session)

**Purpose:** To verify that an iSCSI target ignores any non-immediate commands with a duplicate CmdSN inside of the range specified by MaxCmdSN to ExpCmdSN with multiple connections. In this test the duplicate command is not the first command in a session.

**Reference:** iSCSI Standard Clause 3.2.2.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** January 11, 2008

**Discussion:** For non-immediate commands, the CmdSN field can take any value from ExpCmdSN to MaxCmdSN. The target MUST silently ignore any non-immediate command outside of the range or non-immediate duplicates within the range.

**Test Setup:** The DUT and Test Station pair should be able to make a TCP connection.

**Procedure:**
- Start two connections from the Testing Station to the iSCSI target being tested.
- During login negotiate the following parameters: ImmediateData=No, InitialR2T=Yes, MaxConnections=2.
- Complete the Login Phase and proceed to the Full Feature Phase.
- Issue a SCSI-INQUIRY to the DUT, wait for response data and status.
- Issue a TEST-UNIT-READY to the DUT, wait for response.
- Issue a READ-CAP to the DUT, wait for response data and status.
- On the first connection: issue a series of three SCSI WRITE commands with CmdSNs of X, X+2 and X+2 respectively, each with ExpectedDataTransferLength equal to the MaxRecvDataSegmentLength of the Testing Station. Where X is the ExpCmdSN after the READ-CAP command completes.
- On the second connection: issue a SCSI WRITE command with a CmdSN of X+1 in such a way that the DUT receives this WRITE command after the first three WRITE commands. Where X is the ExpCmdSN after the READ-CAP command completes.
- After all WRITE commands have been transmitted, wait for an R2T for each command and respond to each R2T with a Data-Out PDUs as necessary.

**Observable Results:**
- Verify that the target ignores the command with the duplicate CmdSN value.
- Verify that the target still properly handles the subsequently received WRITE commands.

**Possible Problems:** If the DUT does not support MaxConnections > 1 this item is not testable.
Test #3.2.3 Out of Range CmdSN with Two Connections

**Purpose:** To verify that a device using multiple connections is able to complete SCSI commands on the open connections and properly sets the sequence numbers of each transmitted PDU.

**Reference:** iSCSI Standard Clause 3.2.2.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** February 10, 2009

**Discussion:** For the numbering mechanism, the initiator and target maintain the following three variables for each session:
- **CmdSN** - the current command Sequence Number, advanced by 1 on each command shipped except for commands marked for immediate delivery. CmdSN always contains the number to be assigned to the next Command PDU.
- **ExpCmdSN** - the next expected command by the target. The target acknowledges all commands up to, but not including, this number. The initiator treats all commands with CmdSN less than ExpCmdSN as acknowledged. The target iSCSI layer sets the ExpCmdSN to the largest non-immediate CmdSN that it can deliver for execution plus 1 (no holes in the CmdSN sequence).
- **MaxCmdSN** - the maximum number to be shipped. The queuing capacity of the receiving iSCSI layer is MaxCmdSN - ExpCmdSN + 1. The initiator's ExpCmdSN and MaxCmdSN are derived from target-to-initiator PDU fields.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Open a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=2.
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second connection of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On the first connection, perform two READ commands.
- On the second connection, perform two READ commands.
- On the second connection, transmit a READ Command with an out of range CmdSN.

**Observable Results:**
- Verify that SCSI traffic appears on each connection, and that each SCSI command completes successfully.
- Verify that the DUT silently ignores the command PDU with the out of range CmdSN.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #3.3.1 StatSN Ordering with Two Connections

Purpose: To verify that an iSCSI target using multiple connections is able to properly set the StatSN field.

Reference: iSCSI Standard Clause 3.2.2.2

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: February 18, 2009

Discussion: Responses in transit from the target to the initiator are numbered. The StatSN (Status Sequence Number) is used for this purpose. StatSN is a counter maintained per connection. ExpStatSN is used by the initiator to acknowledge status. The status sequence number space is 32-bit unsigned-integers and the arithmetic operations are the regular mod(2**32) arithmetic. Status numbering starts with the Login response to the first Login request of the connection. The Login response includes an initial value for status numbering (any initial value is valid). Initiators and Targets MUST support the response-numbering scheme.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=2.
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second connection of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On the first connection, perform two READ commands.
- On the second connection, perform two READ commands.

Observable Results:
- Verify that the DUT numbers responses beginning with the first Login Request on each connection.
- Verify that the DUT keeps separate StatSN counters for each connection. The StatSN counter on the second should not 'follow' the counter on the first connection, but rather each counter should be incremented by one each time a response is sent.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #3.4.1 DataSN Ordering with Two Connections

Purpose: To verify that an iSCSI target using multiple connections is able to properly set the DataSN field.

Reference: 3.2.2.3

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 5, 2007

Discussion: Data and R2T PDUs transferred as part of some command execution MUST be sequenced. The DataSN field is used for data sequencing. For input (read) data PDUs, DataSN starts with 0 for the first data PDU of an input command and advances by 1 for each subsequent data PDU. For output data PDUs, DataSN starts with 0 for the first data PDU of a sequence (the initial unsolicited sequence or any data PDU sequence issued to satisfy an R2T) and advances by 1 for each subsequent data PDU.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=2.
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second connection of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On the first connection, perform two READ commands.
- On the second connection, perform two READ commands.

Observable Results:
- Verify that the DUT number Data PDUs for each command starting with 0 and incrementing by 1 for each Data PDU responding to a command.
- Verify that the DUT keeps separate DataSN counters for each command. The DataSN counter on the second command should not 'follow' the counter on the first connection, but rather each counter should be incremented by one each time a Data PDU is sent.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire
InterOperability Laboratory

Test #3.5.1 R2TSN Ordering with Two Connections

Purpose: To verify that an iSCSI target using multiple connections is able to properly set the R2TSN field.

Reference: 3.2.2.3

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 5, 2007

Discussion: Data and R2T PDUs transferred as part of some command execution MUST be sequenced. For example, the first R2T has an R2TSN of 0 and advances by 1 for each subsequent R2T for a command.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a connection of SessionType=Normal from the Testing Station to the DUT. Negotiate MaxConnections=2.
- Complete the Login Phase and proceed to Full Feature Phase.
- Start a second connection of SessionType=Normal to the DUT. Complete the Login Phase and proceed to Full Feature Phase.
- On the first connection, perform two WRITE commands.
- On the second connection, perform two WRITE commands.

Observable Results:
- Verify that the DUT numbers R2T PDUs for each command starting with 0 and incrementing by 1 for each R2T PDU transmitted in response to a command.
- Verify that the DUT keeps separate R2TSN counters for each command. The R2TSN counter on the second command should not 'follow' the counter on the first connection, but rather each counter should be incremented by one each time an R2T PDU is sent.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
GROUP 4: Connection Termination Multiconnection Tests for Targets

Overview: This group of tests verifies the connection termination and task reassignment of multiconnection iSCSI sessions as defined in RFC 3720 and RFC 5048. Comments and questions regarding the implementation of these tests are welcome, and may be forwarded to Peter Scruton, UNH InterOperability Lab (pjs@iol.unh.edu).
Test #4.1.1 Task Reassignment Attempt before DefaultTime2Wait Timer Expires

Purpose: To verify that an iSCSI target properly negotiates and uses DefaultTime2Retain and DefaultTime2Wait when using multiple connections.

Reference: iSCSI Standard Clause 6.4.1, 12.15, 12.16

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: February 18, 2009

Discussion: The DefaultTime2Retain key lets the initiator and target negotiate the maximum time, in seconds after an initial wait (Time2Wait), before which an active task reassignment is still possible after an unexpected connection termination or a connection reset. The DefaultTime2Wait key lets the initiator and target negotiate the minimum time, in seconds, to wait before attempting an explicit/implicit logout or an active task reassignment after an unexpected connection termination or a connection reset.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DefaultTime2Wait=4, DefaultTime2Retain=10, ErrorRecoveryLevel=2.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- Complete the Login Phase of the second connection and proceed to Full Feature Phase operation.
- On each connection, the Testing Station should perform a WRITE command.
- The Testing Station should drop one connection.
- The Testing Station should attempt to reassign the tasks from the dropped connection before the DefaultTime2Wait timer expires.

Observable Results:
- The DUT should not accept the Task Reassignment.

Possible Problems: If the device does not support more than 1 connection, this item is not testable. If the DUT does not support Task Reassignment, this item is Not Testable. Devices may support Task Reassignment without supporting ErrorRecoveryLevel=2; in this case, a lower ErrorRecoveryLevel may be used.
The University of New Hampshire  
InterOperability Laboratory

Test #4.1.2 Task Reassignment Attempt after DefaultTime2Retain Timer Expires

Purpose: To verify that an iSCSI target properly negotiates and uses DefaultTime2Retain and DefaultTime2Wait when using multiple connections.

Reference: iSCSI Standard Clause 6.4.1, 12.15, 12.16

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: February 18, 2009

Discussion: The DefaultTime2Retain key lets the initiator and target negotiate the maximum time, in seconds after an initial wait (Time2Wait), before which an active task reassignment is still possible after an unexpected connection termination or a connection reset. The DefaultTime2Wait key lets the initiator and target negotiate the minimum time, in seconds, to wait before attempting an explicit/implicit logout or an active task reassignment after an unexpected connection termination or a connection reset.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DefaultTime2Wait=4, DefaultTime2Retain=10, ErrorRecoveryLevel=2.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- Complete the Login Phase of the second connection and proceed to Full Feature Phase operation.
- On each connection the Testing Station should perform a WRITE command.
- The Testing Station should drop one connection.
- The Testing Station should attempt to reassign the tasks from the dropped connection after the DefaultTime2Retain timer expires.

Observable Results:
- The DUT should not accept the Task Reassignment.

Possible Problems: If the device does not support more than 1 connection this item is not testable. If the DUT does not support Task Reassignment this item is Not Testable. Devices may support Task Reassignment without supporting ErrorRecoveryLevel=2; in this case, a lower ErrorRecoveryLevel may be used.
The University of New Hampshire
InterOperability Laboratory

Test #4.1.3 Task Reassignment within DefaultTime2Wait and DefaultTime2Retain Range

Purpose: To verify that an iSCSI target properly negotiates and uses DefaultTime2Retain and DefaultTime2Wait when using multiple connections.

Reference: iSCSI Standard Clause 6.4.1, 12.15, 12.16

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: February 18, 2009

Discussion: The DefaultTime2Retain key lets the initiator and target negotiate the maximum time, in seconds after an initial wait (Time2Wait), before which an active task reassignment is still possible after an unexpected connection termination or a connection reset. The DefaultTime2Wait key lets the initiator and target negotiate the minimum time, in seconds, to wait before attempting an explicit/implicit logout or an active task reassignment after an unexpected connection termination or a connection reset.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- Open a session of type "Normal" from the Testing Station to the DUT.
- On the leading login of the session, negotiate DefaultTime2Wait=4, DefaultTime2Retain=10, ErrorRecoveryLevel=2.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Open a new connection within the "Normal" session.
- Complete the Login Phase of the second connection and proceed to Full Feature Phase operation.
- On each connection the Testing Station should perform a WRITE command.
- The Testing Station should drop one connection.
- The Testing Station should attempt to reassign the tasks from the dropped connection after the DefaultTime2Wait timer expires, but before the DefaultTime2Retain timer expires.

Observable Results:
- The DUT should accept the Task Reassignment.

Possible Problems: If the device does not support more than 1 connection this item is not testable. If the DUT does not support Task Reassignment this item is Not Testable. Devices may support Task Reassignment without supporting ErrorRecoveryLevel=2; in this case, a lower ErrorRecoveryLevel may be used.
Test #4.2.1: Logout Response Close Existent Connection

**Purpose:** To see that an iSCSI target properly responds to a received Logout Request.

**Reference:** iSCSI Standard Clause 10.14, 10.15

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** April 5, 2005

**Discussion:** When receiving a Logout request with the reason code of "close the connection" or "close the session", the target MUST terminate all pending commands, whether acknowledged via ExpCmdSN or not, on that connection or session respectively. The target then issues the Logout response and half-closes the TCP connection (sends FIN). After receiving the Logout response and attempting to receive the FIN (if still possible), the initiator MUST completely close the logging-out connection. For the terminated commands, no additional responses should be expected.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
Start 2 connections from the Testing Station to the iSCSI target being tested. On each connection do the following:

- Negotiate the following parameters: ImmediateData=No, InitialR2T=Yes.
- Declare a MaxRecvDataSegmentLength of 1024.
- Perform a standard login and proceed to the Full Feature Phase.
- Issue a SCSI-INQUIRY, TEST-UNIT-READY and READ-CAP to the DUT. Wait for response data and status.
- Issue a WRITE command to the DUT.
- Wait for an R2T from the DUT.
- On one connection only issue a Logout Request with a reason code of 1 'closes the connection'.

**Observable Results:**

- Verify that the DUT issues a Logout Response and FIN only on the connection which the Testing Station issued the Logout Request on. All tasks on the second connection should remain open.
- Verify that the Response field in the Logout Response is set to 0.

**Possible Problems:** None.
Test #4.2.2: Logout Response Close Session

Purpose: To see that an iSCSI target properly responds to a received Logout Request.

Reference: iSCSI Standard Clause 10.14, 10.15

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: April 5, 2005

Discussion: When receiving a Logout request with the reason code of "close the connection" or "close the session", the target MUST terminate all pending commands, whether acknowledged via ExpCmdSN or not, on that connection or session respectively. The target then issues the Logout response and half-closes the TCP connection (sends FIN). After receiving the Logout response and attempting to receive the FIN (if still possible), the initiator MUST completely close the logging-out connection. For the terminated commands, no additional responses should be expected.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
Start 2 connections from the Testing Station to the iSCSI target being tested. On each connection do the following:

- Negotiate the following parameters: ImmediateData=No; InitialR2T=Yes.
- Declare a MaxRecvDataSegmentLength of 1024.
- Perform a standard login and proceed to the Full Feature Phase.
- Issue a SCSI-INQUIRY, TEST-UNIT-READY and READ-CAP to the DUT. Wait for response data and status.
- Issue a WRITE command to the DUT.
- Wait for an R2T from the DUT.
- On one connection only issue a Logout Request with a reason code of 0 'closes the session'.

Observable Results:
- Verify that the DUT half closes each open TCP connection within the session specified to be closed by the Testing Station.
- Verify that the Response field in the Logout Response is set to 0.

Possible Problems: None.
Test #4.2.3: Logout Response Close Nonexistent Connection

Purpose: To see that an iSCSI target properly responds to a received Logout Request.

Reference: iSCSI Standard Clause 10.14, 10.15

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: April 5, 2005

Discussion: When receiving a Logout request with the reason code of "close the connection" or "close the session", the target MUST terminate all pending commands, whether acknowledged via ExpCmdSN or not, on that connection or session respectively. The target then issues the Logout response and half-closes the TCP connection (sends FIN). After receiving the Logout response and attempting to receive the FIN (if still possible), the initiator MUST completely close the logging-out connection. For the terminated commands, no additional responses should be expected.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
Start 2 connections from the Testing Station to the iSCSI target being tested. On each connection do the following:

- Negotiate the following parameters: ImmediateData=No; InitialR2T=Yes.
- Declare a MaxRecvDataSegmentLength of 1024.
- Perform a standard login and proceed to the Full Feature Phase.
- Issue a SCSI-INQUIRY, TEST-UNIT-READY and READ-CAP to the DUT. Wait for response data and status.
- Issue a WRITE command to the DUT.
- Wait for an R2T from the DUT.
- On one connection only issue a Logout Request with a reason code of 1 'closes the connection' for a non-existent CID.

Observable Results:

- Verify that the Response field in the Logout Response is set to 1 = 'CID not found'. Both connections should remain open.

Possible Problems: None.
Overview: This group of tests verifies the task management functionality of multiconnection iSCSI sessions as defined in RFC 3720 and RFC 5048. Comments and questions regarding the implementation of these tests are welcome, and may be forwarded to Peter Scruton, UNH InterOperability Lab (pjs@iol.unh.edu).
The University of New Hampshire  
InterOperability Laboratory

Test #5.1.1: Clarified RFC3720 Multi-Task Abort Semantics

Purpose: To verify that the target properly supports the legacy RFC3720 task reporting model.

Reference: iSCSI Standard Clause 3.5.1.4, 10.6.1, iSCSI Corrections and Clarifications Clause 4.1.2, 9.1

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 13, 2009

Discussion: The Task Management function response carries an indication of function completion for a Task Management function request including how it completed (response and qualifier) and additional information for failure responses. After the Task Management response indicates Task Management function completion, the initiator will not receive any additional responses from the affected tasks. The target iSCSI layer MUST wait for responses on currently valid target-transfer tags of the affected tasks from the issuing initiator, MUST wait for all commands of the affected tasks to be received based on the CmdSN ordering, and MUST propagate the TMF request to and receive the response from the target SCSI layer before sending the Task Management Function response. When TaskReporting is negotiated to RFC3720, response fencing is not guaranteed.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:

• Connect the Testing Station to the iSCSI target being tested.
• Transmit the initial login request to the DUT. This should be the initial request in a Normal session (i.e. not a Discovery session). Try to negotiate TaskReporting=RFC3720.
• Open a second connection from the Testing Station to the DUT.
• On each connection proceed through the Login Phase and in to Full Feature Phase operation.
• On the first connection transmit a SCSI-INQUIRY to the DUT. Wait for response and data from the DUT.
• On the first connection transmit a TEST-UNIT READY to the DUT. Wait for response from the DUT.
• On the first connection transmit a READ Command to the DUT.
• Increment CmdSN twice (should be two greater than the CmdSN of the previous READ Command), and transmit an immediate TMF Request with Function Code 2 'ABORT TASK SET'.
• Decrement CmdSN, (should be 1 more than the previous READ Command) and transmit a READ Command to the DUT on the second connection. This should create an out of order delivery of command PDUs.
• Transmit a non-immediate NOP Out command to the DUT with the same CmdSN as that of the TMF Request.

Observable Results:

• Verify that the DUT responds to the TMF Request with an appropriate TMF Response.
• Verify that the DUT does not respond to the received READ commands after sending the TMF Response.
• Verify that the DUT responds to the received NOP-Out with a NOP-In.

Possible Problems: If ErrorRecoveryLevel = 0, the target may choose to drop the connection in order to complete the ABORT TASK SET command. This may be preceded by an Async Message requesting Logout.
Test #5.2.1: Response Fence Check Prior to Task Management Function Response

**Purpose:** To verify that the target complies with the Response Fence requirements (as specified in RFC 5048) when responding to TMF requests.

**Reference:** iSCSI Standard Clause 3.5.1.4, iSCSI Corrections and Clarifications Clause 3.3.2, 4.1.2, 9.1

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 13, 2009

**Discussion:** The Task Management function response carries an indication of function completion for a Task Management function request including how it completed (response and qualifier) and additional information for failure responses. After the Task Management response indicates Task Management function completion, the initiator will not receive any additional responses from the affected tasks. When TaskReporting is negotiated to ResponseFence, Response Fence semantics MUST be supported in reporting task completions. The Target iSCSI layer MUST provide the Response Fence behavior for the TMF Response on the issuing session as follows:

If it is a multi-connection session, the target iSCSI layer takes note of the last-sent and unacknowledged StatSN on each of the connections in the iSCSI session, and waits for an acknowledgement (NOP-In PDUs MAY be used to solicit acknowledgements as needed in order to accelerate this process) of each such StatSN to clear the fence. The SCSI response requiring Response Fence behavior MUST NOT be sent to the initiator before acknowledgements are received for each of the unacknowledged StatSNs.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Connect the Testing Station to the iSCSI target being tested.
- Transmit the initial login request to the DUT. This should be the initial request in a Normal session (i.e. not a Discovery session). Negotiate TaskReporting=ResponseFence, MaxConnections=3
- Open two more connections from the Testing Station to the DUT.
- On each connection proceed through the Login Phase and into Full Feature Phase operation.
- On the first connection transmit a SCSI INQUIRY to the DUT. Wait for response and data from the DUT.
- On the first connection transmit a TEST UNIT READY to the DUT. Wait for response from the DUT.
- On each connection transmit a READ CAPACITY to the DUT. Wait for data and response from the DUT.
- On the first connection transmit a READ Command to the DUT with CmdSN X.
- Transmit a READ Command with CmdSN (X + 1) to the DUT on the second connection.
- Transmit an immediate TMF Request with CmdSN (X + 2) on the first connection with Function Code 2 'ABORT TASK SET'.
- Wait for 10 seconds.
- On the third connection, send an immediate Nop-Out ping with CmdSN (X + 2) to acknowledge the status from the READ CAPACITY command.
- On the first connection, receive the TMF Response, then transmit a non-immediate NOP Out command to the DUT with CmdSN (X + 2).

**Observable Results:**
- Verify that the DUT responds to the TMF Request with an appropriate TMF Response.
- Verify that the DUT does not respond to the received READ commands after sending the TMF Response.
- Verify that the DUT responds to the received NOP-Outs with NOP-Ins.
- Verify that the DUT does not send the TMF Response until it receives acknowledgement of the READ CAPACITY status on the third connection.

**Possible Problems:** If ErrorRecoveryLevel = 0, the target may choose to drop the connection in order to complete the ABORT TASK SET command. This may be preceded by an Async Message requesting Logout. If the DUT does not
support TaskReporting=ResponseFence, this test is Not Testable. If the DUT does not support MaxConnections>=3, this item is Not Testable.
Test #5.2.2: Response Fence Check After Task Management Function Response

**Purpose:** To verify that the target complies with the Response Fence requirements (as specified in RFC 5048) when responding to TMF Requests.

**Reference:** iSCSI Standard Clause 3.5.1.4, iSCSI Corrections and Clarifications Clause 3.3.2, 4.1.2

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 13, 2009

**Discussion:** The Task Management function response carries an indication of function completion for a Task Management function request including how it completed (response and qualifier) and additional information for failure responses. After the Task Management response indicates Task Management function completion, the initiator will not receive any additional responses from the affected tasks. The DUT MUST provide the Response Fence behavior for the TMF Response on the issuing session as follows:

The target iSCSI layer must wait for an acknowledgement of the SCSI Response PDU that carried the SCSI response requiring the Response Fence behavior. The fence MUST be considered cleared only after receiving the acknowledgement.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- Connect the Testing Station to the iSCSI target being tested.
- Transmit the initial login request to the DUT. This should be the initial request in a Normal session (i.e. not a Discovery session). Negotiate TaskReporting=ResponseFence, MaxConnections=3
- Open two more connections from the Testing Station to the DUT.
- On each connection proceed through the Login Phase and into Full Feature Phase operation.
- On the first connection transmit a SCSI INQUIRY to the DUT. Wait for response and data from the DUT.
- On the first connection transmit a TEST UNIT READY to the DUT. Wait for response from the DUT.
- On each connection transmit a READ CAPACITY to the DUT. Wait for response from the DUT.
- On the first connection transmit a READ Command to the DUT with CmdSN X.
- On the second connection, transmit a READ Command with CmdSN (X + 1) to the DUT.
- On the first connection, transmit an immediate Task Management Function Request with CmdSN (X + 2) with Function Code 2 'ABORT TASK SET'.
- On the third connection, send a non-immediate READ Command with CmdSN (X + 2). This should not be included in the task set to be terminated.
- On the first connection, receive the Task Management Function Response.
- Wait for 10 seconds.
- On the first connection, transmit an immediate NOP Out command to the DUT with CmdSN (X + 3). This should acknowledge the Task Management Function response.

**Observable Results:**
- Verify that the DUT responds to the TMF Request with an appropriate TMF Response.
- Verify that the DUT does not respond to the received READ commands on the first two connections after sending the TMF Response.
- Verify that the DUT responds to the received NOP-Out with a NOP-In.
- Verify that the DUT does not send status for the last READ command until it receives the NOP Out command acknowledging the TMF Response.

**Possible Problems:** If ErrorRecoveryLevel = 0, the target may choose to drop the connection in order to complete the ABORT TASK SET command. This may be preceded by an Async Message requesting Logout. If the DUT does not support TaskReporting=ResponseFence, this test is Not Testable.
Test #5.3.1 FastAbort when Affected Tasks Have Not Been Received

Purpose: To verify that a device properly uses the FastAbort multi-task termination semantics (as specified in RFC 5048).

Reference: iSCSI Corrections and Clarifications Clause 3.3.1, 4.1.3

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 13, 2009

Discussion: Protocol behavior defined in this section MUST be exhibited by iSCSI implementations on an iSCSI session when they negotiate the TaskReporting (Section 9.1) key to "FastAbort" on that session. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of a sequence of actions in a specified order. The target iSCSI layer MUST wait for all commands of the affected tasks to be received based on the CmdSN ordering on the issuing session.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- In the leading login of a session, if not offered by the DUT, offer key=value pairs TaskReporting=FastAbort, InitialR2T=Yes, ImmediateData=No, and MaxConnections=2.
- Open another connection to the DUT.
- Bring both connections into Full Feature Phase.
- Send a WRITE command on each connection of sufficient size that it would take several R2T’s to complete.
- Wait for R2T on each connection and send Data-Outs to fulfill the R2T.
- Wait for another R2T on each connection.
- On one connection, send a Task Management Function request of type ABORT TASK SET, with CmdSN=(CmdSN of second WRITE + 5).

Observable Results:
- Verify that the DUT does not send a TMF Response or an Asynchronous Message with AsyncEvent=5.

Possible Problems: If the DUT does not support TaskReporting=FastAbort, this test is Not Testable. If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire
InterOperability Laboratory

Test #5.3.2 FastAbort with Active Affected Target Transfer Tags

Purpose: To verify that a device properly uses the FastAbort multi-task termination semantics (as specified in RFC 5048).

Reference: iSCSI Corrections And Clarifications Clause 3.3.1, 4.1.3

Resource Requirements: A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

Last Modification: July 13, 2009

Discussion: Protocol behavior defined in this section MUST be exhibited by iSCSI implementations on an iSCSI session when they negotiate the TaskReporting (Section 9.1) key to "FastAbort" on that session. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of a sequence of actions in a specified order. The target iSCSI layer MUST leave all active "affected TTTs" (i.e., active TTTs associated with affected tasks) valid.

Test Setup: The DUT and Test Station pair should be able to make multiple TCP connections.

Procedure:
- In the leading login of a session, if not offered by the DUT, offer key=value pairs TaskReporting=FastAbort, InitialR2T=Yes, ImmediateData=No, and MaxConnections=2.
- Open another connection to the DUT.
- Bring both connections into Full Feature Phase.
- Send a WRITE command on each connection of sufficient size that it would take several R2T’s to complete.
- Wait for R2T on each connection and send Data-Outs to fulfill the R2T.
- Wait for another R2T on each connection.
- On the first connection, send a Task Management Function request of type ABORT TASK SET.
- Send a Data-Out in response to the R2T on each connection.

Observable Results:
- Verify that the DUT does not send a TMF Response.
- Verify that the DUT accepts the Data-Outs and does not send Reject in response to the Data-Outs.

Possible Problems: If the DUT does not support TaskReporting=FastAbort, this test is Not Testable. If the device does not support more than 1 connection this item is not testable.
Test #5.3.3 FastAbort Asynchronous Message Check

**Purpose:** To verify that a device properly uses the FastAbort multi-task termination semantics (as specified in RFC 5048).

**Reference:** iSCSI Corrections And Clarifications Clause 3.3.1, 4.1.3

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 13, 2009

**Discussion:** Protocol behavior defined in this section MUST be exhibited by iSCSI implementations on an iSCSI session when they negotiate the TaskReporting (Section 9.1) key to "FastAbort" on that session. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of a sequence of actions in a specified order. The target iSCSI layer MUST send an Asynchronous Message PDU with AsyncEvent=5 (Section 8.1) on each connection except the issuing connection of the issuing session that has at least one allegiant affected task.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- In the leading login of a session, if not offered by the DUT, offer key=value pairs TaskReporting=FastAbort, InitialR2T=Yes, ImmediateData=No, and MaxConnections=2.
- Open another connection to the DUT.
- Bring both connections into Full Feature Phase.
- Send a WRITE command on each connection of sufficient size that it would take several R2T’s to complete.
- Wait for R2T on each connection and send Data-Outs to fulfill the R2T.
- Wait for another R2T on each connection.
- On the first connection, send a Task Management Function request of type ABORT TASK SET.
- On the second connection, wait for the Asynchronous Message.

**Observable Results:**
- Verify that the DUT does not send a TMF Response.
- Verify that the Asynchronous Message was sent on the second connection and has AsyncEvent=5.
- Verify that an Asynchronous Message with AsyncEvent=5 was not received on the first connection.

**Possible Problems:** If the DUT does not support TaskReporting=FastAbort, this test is Not Testable. If the device does not support more than 1 connection this item is not testable.
Test #5.3.4 FastAbort Multiple LUNs

**Purpose:** To verify that a device properly uses the FastAbort multi-task termination semantics (as specified in RFC 5048).

**Reference:** iSCSI Corrections And Clarifications Clause 3.3.1, 4.1.3

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 13, 2009

**Discussion:** Protocol behavior defined in this section MUST be exhibited by iSCSI implementations on an iSCSI session when they negotiate the TaskReporting (Section 9.1) key to "FastAbort" on that session. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of a sequence of actions in a specified order. The target iSCSI layer MUST send an Asynchronous Message PDU with AsyncEvent=5 (Section 8.1) on each connection except the issuing connection of the issuing session that has at least one allegiant affected task. If there are multiple affected LUs (say, due to a target reset), then one Async Message PDU MUST be sent for each such LU on each connection that has at least one allegiant affected task. The LUN field in the Asynchronous Message PDU MUST be set to match the LUN for each such LU.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections. The DUT should be configured to present at least two LUN’s to the Testing Station.

**Procedure:**
- In the leading login of a session, if not offered by the DUT, offer key=value pairs TaskReporting=FastAbort, InitialR2T=Yes, ImmediateData=No, and MaxConnections=2.
- Open another connection to the DUT.
- Bring both connections into Full Feature Phase.
- Send two WRITE commands on second connection of sufficient size that it would take several R2T’s to complete, each to write to a different LUN.
- Wait for R2T for the first WRITE command and send Data-Outs to fulfill the R2T.
- Wait for another R2T.
- On the first connection, send a Task Management Function request of type TARGET WARM RESET.
- On the second connection, wait for the Asynchronous Message PDUs.

**Observable Results:**
- Verify that the DUT does not send a TMF Response.
- Verify that two Asynchronous Message PDU’s were sent by the DUT on the second connection and have AsyncEvent=5, one for each affected LUN.
- Verify that an Asynchronous Message with AsyncEvent=5 was not received on the first connection.

**Possible Problems:** If the DUT does not support TaskReporting=FastAbort, this item is Not Testable. If the device does not support more than 1 connection this item is Not Testable. The DUT may not support the TARGET WARM RESET TMF Request or may disconnect to fulfill the TARGET WARM RESET TMF Request; in these cases, this item is Not Testable. If the DUT does not present multiple LUNs to the initiator, this item is Not Testable.
Test #5.3.5 FastAbort Response Fence Check

**Purpose:** To verify that a device properly uses the FastAbort multi-task termination semantics (as specified in RFC 5048).

**Reference:** iSCSI Corrections and Clarifications Clause 3.3.1, 4.1.3

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 13, 2009

**Discussion:** Protocol behavior defined in this section MUST be exhibited by iSCSI implementations on an iSCSI session when they negotiate the TaskReporting (Section 9.1) key to "FastAbort" on that session. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of a sequence of actions in a specified order. The target iSCSI layer MUST address the Response Fence flag on the TMF Response on the issuing session as defined in Section 3.3.2.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections.

**Procedure:**
- In the leading login of a session, if not offered by the DUT, offer key=value pairs TaskReporting=FastAbort, InitialR2T=Yes, ImmediateData=No, and MaxConnections=2.
- Open another connection to the DUT.
- Bring both connections into Full Feature Phase.
- Send a WRITE command on each connection of sufficient size that it would take several R2T’s to complete.
- Wait for R2T on each connection and send Data-Outs to fulfill the R2T.
- Wait for another R2T on each connection.
- On the first connection, send a Task Management Function request of type ABORT TASK SET.
- Wait for an Asynchronous Message PDU on the second connection with AsyncEvent=5.
- Send a Nop-Out with ITT=0xffffffff and LUN=(LUN of AsyncMsg PDU) on the second connection.
- Wait for the Task Management Function Response. If any other response PDUs are received, send a Nop-Out ping to acknowledge the StatSN.

**Observable Results:**
- Verify that if the DUT sends SCSI Responses for the WRITE commands, they are sent before the TMF Response.
- Verify that after the acknowledgement of the Asynchronous Message the DUT sends a properly formed TMF Response.

**Possible Problems:** If the DUT does not support TaskReporting=FastAbort, this test is Not Testable. If the device does not support more than 1 connection this item is not testable.
The University of New Hampshire
InterOperability Laboratory

Test #5.4.1 FastAbort Logical Unit Reset with FastAbort Third-Party Session

**Purpose:** To verify that a device properly uses the FastAbort multi-task termination semantics (as specified in RFC 5048) with regard to third-party sessions.

**Reference:** iSCSI Corrections and Clarifications Clause 3.3.2, 4.1.3, 4.1.4

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 14, 2009

**Discussion:** Protocol behavior defined in RFC 5048 Section 4.1.3 MUST be exhibited by iSCSI implementations on an iSCSI session when they negotiate the TaskReporting (Section 9.1) key to "FastAbort" on that session. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of a sequence of actions in a specified order. If the TaskReporting key is negotiation to FastAbort, the target iSCSI layer MUST leave all active "affected TTTs" (i.e., active TTTs associated with affected tasks) valid.

In addition, the target iSCSI layer MUST address the Response Fence flag on the first post-TMF Response on third-party sessions as defined in Section 3.3.2 of RFC 5048. In particular, the target iSCSI layer must wait for an acknowledgement of the SCSI Response PDU that carried the SCSI response requiring the Response Fence behavior. The fence MUST be considered cleared only after receiving the acknowledgement. All further status processing for the LU is resumed only after clearing the fence. If any new responses for the I_T_L nexus are received from the SCSI layer before the fence is cleared, those Response PDUs MUST be held and queued at the iSCSI layer until the fence is cleared.

If the third-party affected session is a FastAbort session and the issuing session is a FastAbort session, the initiator in the third-party role MUST respond to each Async Message PDU with AsyncEvent=5 as defined in Section 8.1. Note that an initiator MAY thus receive these Async Messages on a third-party affected session even if the session is a single-connection session.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections and multiple iSCSI sessions. The DUT should be configured to present the same LUN (referred to in this test as LUN A) to both sessions.

**Procedure:**
- Open two sessions to the DUT. In the leading login of each session, if not offered by the DUT, offer key=value pairs TaskReporting=FastAbort, InitialR2T=Yes, and ImmediateData=No, and MaxConnections=2.
- On each session, open a second connection to the DUT.
- On all connections of all sessions:
  - Enter Full Feature Phase. Send INQUIRY and TEST UNIT READY for LUN A on one connection of each session, wait for any response data and status.
  - Send READ CAPACITY for LUN A. Wait for response data and status.
  - Send a WRITE command of sufficient size for LUN A that it would take several R2T’s to complete.
  - Wait for R2T and send one Data-Out in response.
- On the first connection of the first session, send a TMF Request of type LOGICAL UNIT RESET for LUN A. On all other connections of all sessions, wait for an Asynchronous Message PDU with AsyncEvent=5 and LUN=A.
- On the second connection of the first session, send a Nop-Out with ITT=0xffffffff and LUN=A.
- On the first connection of the first session, wait for the TMF Response. If any other response PDUs are received on the first session prior to the TMF Response, send a Nop-Out ping to acknowledge the StatSN.
- On the second session:
  - On each connection, send an additional Data-Out PDU for the respective WRITE command.
  - On each connection, send a Nop-Out PDU with ExpStatSN=(StatSN of Asynchronous Message PDU + 1), ITT=0xffffffff, and LUN=A.
  - On the first connection, send a TEST UNIT READY for LUN A command to receive the Unit Attention status from the Logical Unit Reset.
  - On the second connection, send another TEST UNIT READY for LUN A.
On the first connection, send a Nop-Out PDU acknowledging the StatSN of the TEST UNIT READY command.

**Figure 1. Diagram of test #5.4.1 procedure**

**Observable Results:**
- Verify that if the DUT sends SCSI Responses for the WRITE commands on the first session, they are sent before the TMF Response.
- Verify that the DUT sends Asynchronous Message PDU’s on all sessions and all connections other than the connection that issued the TMF Request.
- Verify that after the acknowledgement of the Asynchronous Message on the second connection of the first session the DUT sends a properly formed TMF Response on the first connection of the first session.
- Verify that the DUT accepts Data-Out PDU’s on the second session even after the TMF Response has been received on the first session.
- Verify that the DUT exhibits the Response Fence behavior for the first TEST UNIT READY response carrying the Unit Attention by waiting to send the next SCSI Response until the first status is acknowledged via the Nop-Out PDU.
Possible Problems: If the DUT does not support TaskReporting=FastAbort, this test is Not Testable. If the device does not support more than 1 connection this item is not testable. The WRITE commands should be for different locations on the disk to avoid conflicts between the sessions and connections.
Test #5.4.2 FastAbort Logical Unit Reset with RFC3720 Third-Party Session

**Purpose:** To verify that a device properly uses the FastAbort multi-task termination semantics (as specified in RFC 5048) with regard to third-party sessions.

**Reference:** iSCSI Corrections and Clarifications Clause 4.1.3, 4.1.4

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 14, 2009

**Discussion:** Protocol behavior defined in RFC 5048 Section 4.1.3 MUST be exhibited by iSCSI implementations on an iSCSI session when they negotiate the TaskReporting (Section 9.1) key to "FastAbort" on that session. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of a sequence of actions in a specified order. If the issuing session is a FastAbort session, the iSCSI target implementation is FastAbort-capable, and the third-party affected session is an RFC 3720 session, the following behavior MUST be exhibited by the iSCSI target layer: Asynchronous Message PDUs MUST NOT be sent on the third-party session to prompt the FastAbort behavior.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections and multiple iSCSI sessions. The DUT should be configured to present the same LUN (referred to in this test as LUN A) to both sessions.

**Procedure:**

- Open two connections to the DUT. In the leading login of each session, if not offered by the DUT, offer key=value pairs InitialR2T=Yes, and ImmediateData=No. On the first session, offer TaskReporting=FastAbort, MaxConnections=2. On the second session, offer TaskReporting=RFC3720.
- On the first session, open a second connection to the DUT.
- On all connections of all sessions:
  - Enter Full Feature Phase. Send INQUIRY and TEST UNIT READY for LUN A on one connection of each session, wait for any response data and status.
  - Send READ CAPACITY for LUN A. Wait for response data and status.
  - Send a WRITE command for LUN A of sufficient size that it would take several R2T’s to complete.
  - Wait for R2T with LUN=A and send one Data-Out in response.
- On the first connection of the first session, send a TMF Request of type LOGICAL UNIT RESET for LUN A.
- On the second connection of the first session:
  - Wait for an Asynchronous Message PDU with AsyncEvent=5 and LUN=A.
  - Send a Nop-Out with ITT=0xffffffff and LUN=A.
- On the second session, complete the data transfer for the R2T. Receive any SCSI Response that may be sent by the DUT.
- On the first connection of the first session, wait for the TMF Response. If any other response PDUs are received on the first session prior to the TMF Response, send a Nop-Out ping to acknowledge the StatSN.
Observable Results:
- Verify that if the DUT sends SCSI Responses for the WRITE commands on the first session, they are sent before the TMF Response.
Verify that the DUT sends properly formed Asynchronous Message PDU’s on all connections of the first session other than the connection that issued the TMF Request.

Verify that the DUT did not send an Asynchronous Message on the second session where TaskReporting was negotiated to RFC3720.

Verify that after the acknowledgement of the Asynchronous Message on the first session, the DUT sends a properly formed TMF Response.

Verify that the DUT does not send any SCSI Responses for the WRITE commands after the reception of the TMF Response.

Possible Problems: If the DUT does not support TaskReporting=FastAbort, this test is Not Testable. If the device does not support more than 1 connection this item is not testable. The WRITE commands should be for different locations on the disk to avoid conflicts between the sessions and connections. The DUT is not required to wait for all TTT’s on third-party sessions to complete before sending the TMF Response.
Test #5.4.3 RFC3720 Logical Unit Reset with FastAbort Third-Party Session

**Purpose:** To verify that a device properly uses the FastAbort multi-task termination semantics (as specified in RFC 5048) with regard to third-party sessions.

**Reference:** iSCSI Corrections and Clarifications Clause 4.1.3, 4.1.4

**Resource Requirements:** A Test Generator tool capable of producing iSCSI PDUs and transporting them over a TCP connection.

**Last Modification:** July 14, 2009

**Discussion:** Protocol behavior defined in this section MUST be exhibited by iSCSI implementations on an iSCSI session when they negotiate the TaskReporting (Section 9.1) key to "FastAbort" on that session. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of a sequence of actions in a specified order. On sessions with TaskReporting=RFC3720 or TaskReporting=ResponseFence, the target iSCSI layer MUST wait for responses on currently valid target-transfer tags of the affected tasks from the issuing initiator. On sessions with TaskReporting=FastAbort, the target iSCSI layer MUST leave all active "affected TTTs" (i.e., active TTTs associated with affected tasks) valid.

If the issuing session is an RFC 3720 session, the iSCSI target implementation is FastAbort-capable, and the third-party affected session is a FastAbort session, the following behavior SHOULD be exhibited by the iSCSI target layer: between Steps c and d of the target behavior in Section 4.1.2, send an Asynchronous Message PDU with AsyncEvent=5 (Section 8.1) on each connection of each third-party session to which at least one affected task is allegiant.

**Test Setup:** The DUT and Test Station pair should be able to make multiple TCP connections and multiple iSCSI sessions. The DUT should be configured to present the same LUN (referred to in this test as LUN A) to both sessions.

**Procedure:**

- Open two connections to the DUT. In the leading login of each session, if not offered by the DUT, offer key=value pairs InitialR2T=Yes, and ImmediateData=No. On the first session, offer TaskReporting=RFC3720, MaxConnections=2. On the second session, offer TaskReporting=FastAbort.
- On the first session, open a second connection to the DUT.
- On all connections of all sessions:
  - Enter Full Feature Phase. Send INQUIRY and TEST UNIT READY for LUN A on one connection of each session, wait for any response data and status.
  - Send a WRITE command for LUN A of sufficient size that it would take several R2T’s to complete.
- On the first connection of the first session, send a TMF Request of type LOGICAL UNIT RESET for LUN A.
- On the second session:
  - Wait for an Asynchronous Message PDU with AsyncEvent=5 and LUN=A.
  - Send a Data-Out PDU for the WRITE command.
  - Send a Nop-Out with ITT=0xffffffff and LUN=A.
- On each connection of the first session, complete the data transfer for the current R2T.
- On the first connection of the first session, wait for the TMF Response. If any other response PDUs are received on the first session prior to the TMF Response, send a Nop-Out ping to acknowledge the StatSN.
**Observable Results:**
- Verify that if the DUT sends SCSI Responses for the WRITE commands on the first session, they are sent before the TMF Response.
The University of New Hampshire  
InterOperability Laboratory  

- Verify that the DUT sends an Asynchronous Message PDU with AsyncEvent=5 and LUN=A on the second session. This observable result is recommended behavior according to RFC 5048 section 4.1.4, but it is not a requirement.
- Verify that the DUT does not send an Asynchronous Message PDU with AsyncEvent=5 on the first session.
- Verify that after the Data-Out sequences have been received by the DUT on the first session, the DUT sends a properly formed TMF Response on the first session.
- Verify that the DUT does not send the TMF Response until the data transfers for the R2T’s on the first session have been completed.

Possible Problems: If the DUT does not support TaskReporting=FastAbort, this test is Not Testable. If the device does not support more than 1 connection this item is not testable. The WRITE commands should be for different locations on the disk to avoid conflicts between the sessions and connections.