MODIFICATION RECORD

1. Currently on Version 0.1.
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
INTRODUCTION

Overview
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 suite of tests has been developed to help implementers evaluate the functioning of their iSCSI products. The tests do not determine if a product conforms to the iSCSI draft standard, nor are they purely interoperability tests. Rather, they provide one method to isolate problems within an iSCSI device. Successful completion of all tests contained in this suite does not guarantee that the tested device will operate with other iSCSI devices. However, combined with satisfactory operation in the IOL’s semi-production environment, these tests provide a reasonable level of confidence that the Device Under Test (DUT) will function well in most multivendor iSCSI environments.

Organization of Tests
The tests contained in this document are organized to simplify the identification of information related to a test and to facilitate in the actual testing process. Each test contains an identification section that describes the test and provides cross reference information. The detailed section discusses the background information and specifies how the test is to be performed. Tests are grouped in order to reduce setup time in the lab environment. Each test contains the following information:

Test Label
The Label associated with each test is a title that is used to refer to the test. The attached number is an internal reference number dealing with an internal reference to the test.

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

References
The references section lists cross references to the iSCSI draft standard and other documentation that might be helpful in understanding and evaluating the test and results.

Resource Requirements
The requirements section specifies the software, hardware, and test equipment that will be needed to perform the test. The items contained in this section are special test devices, software that must reside on the DUT, or other facilities which may not be available on all devices.
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.

Test Setup
The setup section describes in detail the configuration of the test environment and includes a block diagram for clarification as well as information such as the interconnection of devices, what monitoring equipment should capture, what the generation equipment should send, and any other configuration information vital to carrying out the test. Small changes in the configuration should be included in the test procedure.

Procedure
The procedure section of the test description contains the step-by-step instructions for carrying out the test. It provides a cookbook approach to testing, and will often be interspersed with observable results.

Observable Results
The observable results section lists observables that can be examined by the tester to verify that the DUT is operating properly. When multiple values are possible for an observable, this section provides a short discussion on how to interpret them. Note that complete delineation between the observables in the Procedure and Observable Results is virtually impossible. As such a careful note should be made of the requirements in both sections. In certain cases, it may be necessary to modify certain steps in the Procedure section while doing the actual tests so as to be able to perform the tests. In such cases, the modifications will be noted in the summary report.

Possible Problems
This section provides some clues to look for if the test does not yield the expected results.
REFERENCES

The following documents are referenced in this text:

IETF IPS Working Group iSCSI draft 20
TEST SETUPS

The following test setups are used in this test suite:

Test Setup 1:

Testing Station/ Monitor  DUT
# TABLE OF CONTENTS

- MODIFICATION RECORD ................................................................. 2
- ACKNOWLEDGMENTS ................................................................. 3
- INTRODUCTION ............................................................................ 4
- REFERENCES ............................................................................... 6
- TEST SETUPS ............................................................................... 7
- TABLE OF CONTENTS ..................................................................... 8
- Test #1.1 Basic READ ................................................................. 10
- Test #1.2 Basic READ ................................................................. 11
- Test #1.3 Basic WRITE ............................................................... 12
- Test #1.4 Basic WRITE ............................................................... 13
- Test #2.1 MaxConnections ........................................................ 14
- Test #2.2 MaxConnections ........................................................ 15
- Test #2.3 MaxConnections ........................................................ 16
- Test #3.1 TargetName ................................................................. 17
- Test #4.1 InitiatorName ............................................................... 18
- Test #5.1 InitiatorAlias ............................................................... 19
- Test #6.1 InitialR2T ................................................................. 20
- Test #6.2 InitialR2T ................................................................. 21
- Test #6.3 InitialR2T ................................................................. 22
- Test #6.4 InitialR2T ................................................................. 23
- Test #6.5 InitialR2T ................................................................. 24
- Test #6.6 InitialR2T ................................................................. 25
- Test #7.1 ImmediateData .......................................................... 26
- Test #7.2 ImmediateData .......................................................... 27
- Test #7.3 ImmediateData .......................................................... 28
- Test #7.4 ImmediateData .......................................................... 29
- Test #7.5 ImmediateData .......................................................... 30
- Test #7.6 ImmediateData .......................................................... 31
- Test #8.1 MaxRecvDataSegmentLength ...................................... 32
- Test #8.2 MaxRecvDataSegmentLength ...................................... 33
- Test #8.3 MaxRecvDataSegmentLength ...................................... 34
- Test #8.4 MaxRecvDataSegmentLength ...................................... 35
- Test #9.1 MaxBurstLength .......................................................... 36
- Test #9.2 MaxBurstLength .......................................................... 37
- Test #10.1 FirstBurstLength ......................................................... 38
- Test #10.2 FirstBurstLength ......................................................... 39
- Test #11.1 DefaultTime2Wait .................................................... 40
- Test #11.2 DefaultTime2Wait .................................................... 41
- Test #12.1 DefaultTime2Retain ................................................... 42
- Test #12.2 DefaultTime2Retain ................................................... 43
Test #13.1 Connection Terminated ................................................................. 44
Test #14.1 MaxOutstandingR2T........................................................................ 45
Test #14.2 MaxOutstandingR2T........................................................................ 46
Test #14.3 MaxOutstandingR2T........................................................................ 47
Test #14.4 MaxOutstandingR2T........................................................................ 48
Test #15.1 DataPDUInOrder ........................................................................... 49
Test #15.2 DataPDUInOrder ........................................................................... 50
Test #15.3 DataPDUInOrder ........................................................................... 51
Test #15.4 DataPDUInOrder ........................................................................... 52
Test #15.5 DataPDUInOrder ........................................................................... 53
Test #15.6 DataPDUInOrder ........................................................................... 54
Test #16.1 DataSequenceInOrder ................................................................... 55
Test #16.2 DataSequenceInOrder ................................................................... 56
Test #16.3 DataSequenceInOrder ................................................................... 57
Test #16.4 DataSequenceInOrder ................................................................... 58
Test #16.5 DataSequenceInOrder ................................................................... 59
Test #16.6 DataSequenceInOrder ................................................................... 60
Test #17.1 HeaderDigests and DataDigests ...................................................... 61
Test #17.2 HeaderDigests and DataDigests ...................................................... 62
Test #18.1 Closed Window ........................................................................... 63
Test #1.1 Basic READ

**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:** Mon Jun 2 13:13:32 2003

**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:**
- Allow the DUT to start a connection of SessionType=Normal to the Testing Station. 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 the DUT starts CmdSN numbering with the first Login Request in the leading connection, and increments CmdSN for every Command PDU thereafter.
- Verify that CmdSN is maintained as a Session-wide counter, and does not start back at 0 on the first Login Request of the second connection but is incremented by 1 from the last Login Request or Command PDU sent on the first connection.
- Verify that on both connections that CmdSN is incremented properly. Verify that no duplicate or out of range CmdSNs are transmitted.
- 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 Basic READ

**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:** Mon Jun 23 15:07:05 2003

**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 at least 4 TCP connections.

**Procedure:**
- Allow the DUT to start a connection of SessionType=Normal to the Testing Station. 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 the DUT starts CmdSN numbering with the first Login Request in the leading connection, and increments CmdSN for every Command PDU thereafter.
- Verify that CmdSN is maintained as a Session-wide counter, and does not start back at 0 on the first Login Request of each new connection but is incremented by 1 from the last Login Request or Command PDU sent on the previous connection.
- Verify that on each connection that CmdSN is incremented properly. Verify that no duplicate or out of range CmdSNs are transmitted.
- 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 #1.3 Basic WRITE

**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:** Mon Jun 2 13:14:50 2003

**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:**
- Allow the DUT to start a connection of SessionType=Normal to the Testing Station. 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 the DUT starts CmdSN numbering with the first Login Request in the leading connection, and increments CmdSN for every Command PDU thereafter.
- Verify that CmdSN is maintained as a Session-wide counter, and does not start back at 0 on the first Login Request of the second connection but is incremented by 1 from the last Login Request or Command PDU sent on the first connection.
- Verify that on both connections that CmdSN is incremented properly. Verify that no duplicate or out of range CmdSNs are transmitted.
- 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 #1.4 Basic WRITE

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: Mon Jun 23 15:06:54 2003

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 at least 4 TCP connections.

Procedure:
· Allow the DUT to start a connection of SessionType=Normal to the Testing Station. 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 the DUT starts CmdSN numbering with the first Login Request in the leading connection, and increments CmdSN for every Command PDU thereafter.
· Verify that CmdSN is maintained as a Session-wide counter, and does not start back at 0 on the first Login Request of each new connection but is incremented by 1 from the last Login Request or Command PDU sent on the previous connection.
· Verify that on each connection that CmdSN is incremented properly. Verify that no duplicate or out of range CmdSNs are transmitted.
· 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 #2.1 MaxConnections

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

**Reference:** 12.2

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

**Last Modification:** Mon Feb 10 13:30:46 2003

**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:**
- In the leading login of a session, allow the DUT to declare a value for MaxConnections greater than 1. Negotiate a value of 1 if the DUT supports 2 connections. Negotiate a value of 2 if the DUT supports more than 2 connections.

**Observable Results:**
- Verify that the DUT does not attempt to open any more connections to the Testing Station than the 2 allowed by MaxConnections.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #2.2 MaxConnections

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

**Reference:** 12.2

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

**Last Modification:** Thu Jun 5 15:51:09 2003

**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:**
- In the leading login of a session, allow the DUT to declare a value for MaxConnections greater than 1.
- Allow the DUT to open a second connection with the Testing Station.

**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 #2.3 MaxConnections

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

**Reference:** 5.3, 12.2

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

**Last Modification:** Thu Jun 5 16:08:24 2003

**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:**
- In the leading login of a session, allow the DUT to declare a value for MaxConnections greater than 1.
- Allow the DUT to open a second connection with the Testing Station. Offer a value of 4 for MaxConnections. This should appear as a re-negotiation of MaxConnections to the DUT.

**Observable Results:**
- Verify that in the second connection the DUT recognizes the re-negotiation and drops the second connection.

**Possible Problems:** If the device does not support more than 1 connection.
Test #3.1 TargetName

**Purpose:** To verify that a device receiving an iSCSI TargetName uses it on every connection within a session.

**Reference:** 5.3, 11.3

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

**Last Modification:** Mon Apr 28 10:45:01 2003

**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:**
- Allow the DUT to request the TargetName via SendTargets during a Discovery session.
- Allow the DUT to start a new session of SessionType="Normal".
- In this new "Normal" session negotiate a value for MaxConnections greater than 1.
- Allow the DUT to open a second connection within this Normal session with the Testing Station.

**Observable Results:**
- Verify that the DUT uses the same TargetName key=value pair in the first login request of each connection.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #4.1 InitiatorName

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

**Reference:** 12.5

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

**Last Modification:** Mon Apr 28 10:45:17 2003

**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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login 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 first Login Request in each connection contains the same InitiatorName key.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #5.1 InitiatorAlias

**Purpose:** To verify that an iSCSI initiator uses an InitiatorAlias properly.

**Reference:** 12.7

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

**Last Modification:** Mon Jun 30 08:43:13 2003

**Discussion:** If an initiator has been configured with a human-readable name or description, it SHOULD be communicated to the target during a Login Request PDU. If not, the host name can be used instead. This string is not used as an identifier, nor is meant to be used for authentication or authorization decisions. It can be displayed by the target's user interface in a list of initiators to which it is connected.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- 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 InitiatorAlias key in a Login Request PDU if the DUT was configured with an InitiatorAlias.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable. If the DUT does not transmit the InitiatorAlias key this item is not testable.
Test #6.1 InitialR2T

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

Reference: 12.10

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

Last Modification: Mon Feb 10 13:34:12 2003

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:
- Allow the DUT to open a session of type "Normal".
- 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 complete the Login Phase, and proceed to Full Feature Phase.
- On each connection allow the DUT to 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 any data associated with the WRITE command until R2T is received from the Testing Station.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #6.2 InitialR2T

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

**Reference:** 12.10

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

**Last Modification:** Mon Feb 10 13:34:36 2003

**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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate InitialR2T=No.
- 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 complete the Login Phase, and proceed to Full Feature Phase.
- On each connection allow the DUT to 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 wait for R2T before transmitting 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 #6.3 InitialR2T

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

**Reference:** 5.3, 12.10

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

**Last Modification:** Mon Feb 10 14:10:14 2003

**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. If a renegotiation is detected by an Initiator, the Initiator must drop the connection. A renegotiation must be detected by the Initiator.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- 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=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.
Test #6.4 InitialR2T

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

**Reference:** 5.3, 12.10

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

**Last Modification:** Mon Feb 10 14:10:39 2003

**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. If a renegotiation is detected by an Initiator, the Initiator must drop the connection. A renegotiation must be detected by the Initiator.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- 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 drops the connection.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #6.5 InitialR2T

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

**Reference:** 5.3, 12.10

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

**Last Modification:** Mon Feb 10 14:11:06 2003

**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. If a renegotiation is detected by an Initiator, the Initiator must drop the connection. A renegotiation must be detected by the Initiator.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate InitialR2T=No.
- 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 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 #6.6 InitialR2T

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

**Reference:** 5.3, 12.10

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

**Last Modification:** Mon Feb 10 14:11:36 2003

**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. If a renegotiation is detected by an Initiator, the Initiator must drop the connection. A renegotiation must be detected by the Initiator.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate InitialR2T=No.
- 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=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 #7.1 ImmediateData

**Purpose:** To verify that an iSCSI initiator 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:** Mon Feb 10 14:15:34 2003

**Discussion:** The ImmediateData key determines if the target will accept ImmediateData in non-Data PDUs. ImmediateData is defined as a Session Wide parameter.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate ImmediateData=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 complete the Login Phase, and proceed to Full Feature Phase.
- On each connection allow the DUT to 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 transmits 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 #7.2 ImmediateData

**Purpose:** To verify that an iSCSI initiator 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:** Mon Feb 10 14:15:52 2003

**Discussion:** The ImmediateData key determines if the target will accept ImmediateData in non-Data PDUs. ImmediateData is defined as a Session Wide parameter.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate ImmediateData=No.
- 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 complete the Login Phase, and proceed to Full Feature Phase.
- On each connection allow the DUT to 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 does not transmit ImmediateData with the WRITE command.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #7.3 ImmediateData

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

**Reference:** 5.3, 12.11

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

**Last Modification:** Mon Feb 10 14:16:13 2003

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

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate ImmediateData=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 the Testing Station should offer the key=value ImmediateData=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 ImmediateData=Yes this item is not testable.
Test #7.4 ImmediateData

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

Reference: 5.3, 12.11

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

Last Modification: Mon Feb 10 14:36:44 2003

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

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

Procedure:
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session, negotiate ImmediateData=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 the Testing Station should offer the key=value ImmediateData=No.

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 ImmediateData=Yes this item is not testable.
Test #7.5 ImmediateData

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

**Reference:** 5.3, 12.11

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

**Last Modification:** Mon Feb 10 14:37:00 2003

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

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

**Procedure:**
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session, negotiate ImmediateData=No.
· 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 the Testing Station should offer the key=value ImmediateData=No.

**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.
Test #7.6 ImmediateData

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

**Reference:** 5.3, 12.11

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

**Last Modification:** Mon Feb 10 14:37:15 2003

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

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

**Procedure:**
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session, negotiate ImmediateData=No.
· 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 the Testing Station should offer the key=value ImmediateData=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.
Test #8.1 MaxRecvDataSegmentLength

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

**Reference:** 12.12

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

**Last Modification:** Mon Feb 10 14:40:42 2003

**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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, declare MaxRecvDataSegmentLength=512.
- 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 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 the DUT should attempt to perform a WRITE operation.

**Observable Results:**
- Verify that on the first connection the DUT uses the declared value of 512 for MaxRecvDataSegmentLength.
- Verify that on the second connection the DUT does not use the value of 512 for MaxRecvDataSegmentLength.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #8.2 MaxRecvDataSegmentLength

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

**Reference:** 12.12

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

**Last Modification:** Mon Feb 10 14:42:50 2003

**Discussion:** The ImmediateData 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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, not not declare a value for MaxRecvDataSegmentLength.
- 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 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 DUT should attempt to perform a WRITE operation.

**Observable Results:**
- Verify that on the first connection the DUT does not use the value of 512 for MaxRecvDataSegmentLength.
- Verify that on the second connection the DUT uses the declared value of 512 for MaxRecvDataSegmentLength.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #8.3 MaxRecvDataSegmentLength

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

**Reference:** 12.12

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

**Last Modification:** Mon Feb 10 14:43:04 2003

**Discussion:** The ImmediateData 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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, declare MaxRecvDataSegmentLength=512.
- 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 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 DUT should attempt to perform a WRITE operation.

**Observable Results:**
- Verify that on the first connection the DUT uses the declared value of 512 for MaxRecvDataSegmentLength.
- Verify that on the second connection the DUT uses the value of 1024 for MaxRecvDataSegmentLength.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #8.4 MaxRecvDataSegmentLength

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

Reference: 12.12

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

Last Modification: Mon Feb 10 14:43:17 2003

Discussion: The ImmediateData 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:
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session, declare MaxRecvDataSegmentLength=1024.
· 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 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 WRITE operation.

Observable Results:
· Verify that on the first connection the DUT uses the declared value of 1024 for MaxRecvDataSegmentLength.
· Verify that on the second connection the DUT uses the value of 512 for MaxRecvDataSegmentLength.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #9.1 MaxBurstLength

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

**Reference:** 12.13

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

**Last Modification:** Mon Feb 10 14:51:28 2003

**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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session negotiate MaxBurstLength=512.
- 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 complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection the DUT should attempt to perform a WRITE 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-Out with the F-bit set.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #9.2 MaxBurstLength

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

**Reference:** 5.3, 12.13

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

**Last Modification:** Mon Feb 10 14:51:50 2003

**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 initiator must detect renegotiations, and if one occurs, drop the connection.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session negotiate MaxBurstLength=512.
- 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 the Testing Station should attempt to negotiate MaxBurstLength=512.

**Observable Results:**
- Verify that 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.
Test #10.1 FirstBurstLength

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

**Reference:** 12.14

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

**Last Modification:** Mon Feb 10 14:52:11 2003

**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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session negotiate FirstBurstLength=512.
- 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 complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection the DUT should attempt to perform a WRITE operation.

**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 uses the value of 512 for FirstBurstLength, not transmitting more than 512 bytes of unsolicited data either as immediate data or in unsolicited data PDUs.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #10.2 FirstBurstLength

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

Reference: 5.3, 12.14

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

Last Modification: Mon Feb 10 14:52:32 2003

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 initiator must detect renegotiations, and if one occurs, drop the connection.

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

Procedure:
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session negotiate FirstBurstLength=512.
· 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 the Testing Station should attempt to negotiate FirstBurstLength=512.

Observable Results:
· Verify that 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.
Test #11.1 DefaultTime2Wait

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

**Reference:** 12.15

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

**Last Modification:** Mon Feb 10 14:52:48 2003

**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:**
- Allow the DUT to 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.
- Allow the DUT to 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 #11.2 DefaultTime2Wait

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

**Reference:** 5.3, 12.15

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

**Last Modification:** Mon Feb 10 14:53:22 2003

**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 an initiator detects a renegotiation it must drop the connection. Renegotiations must be detected if they occur.

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

**Procedure:**
- Allow the DUT to 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.
- Allow the DUT to 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 drops the connection.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #12.1 DefaultTime2Retain

Purpose: To verify that an iSCSI initiator 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: Mon Feb 10 14:53:48 2003

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:
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session negotiate DefaultTime2Retain=10.
· Complete the Login Phase and proceed to Full Feature Phase operation.
· Allow the DUT to 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 #12.2 DefaultTime2Retain

**Purpose:** To verify that an iSCSI initiator 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:** Mon Feb 10 14:54:06 2003

**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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session negotiate DefaultTime2Retain=10.
- 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 of the session negotiate DefaultTime2Retain=5.

**Observable Results:**
- Verify that 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.
Test #13.1 Connection Terminated

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

**Reference:** 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:** Mon Feb 10 14:54:37 2003

**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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session negotiate DefaultTime2Wait=4, DefaultTime2Retain=10.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Allow the DUT to 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 initiator should perform a WRITE command.
- The Testing Station should drop one connection

**Observable Results:**
- Verify that the DUT does not attempt to explicitly logout the closed connection or reassign the tasks from that connection outside of the window defined by by DefaultTime2Wait and DefaultTime2Retain. This window goes starts at DefaultTime2Wait after the connection is dropped, and ends at DefaultTime2Wait + DefaultTime2Retain after the connection is dropped.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable. The DUT is not required to attempt to explicitly logout the connection or reassign the task, however it does choose to do so, this must occur during the defined window.
Test #14.1 MaxOutstandingR2T

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

**Reference:** 12.17

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

**Last Modification:** Tue Feb 11 08:21:48 2003

**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. Initiators are required to detect renegotiations, and drop the connection when renegotiations occur.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session negotiate MaxOutstandingR2T=4.
- Complete the Login Phase and proceed to Full Feature Phase operation.
- Allow the DUT to 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 #14.2 MaxOutstandingR2T

**Purpose:** To verify that an iSCSI initiator 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:** Tue Feb 11 08:22:04 2003

**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. Initiators are required to detect renegotiations, and drop the connection when renegotiations occur.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading connection of the session negotiate MaxOutstandingR2T=4.
- 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 of the session negotiate MaxOutstandingR2T=4.

**Observable Results:**
- Verify that the DUT detects the renegotiation and drops the connection.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #14.3 MaxOutstandingR2T

**Purpose:** To verify that an iSCSI initiator 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:** Mon Jun 30 08:42:17 2003

**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. Initiators are required to detect renegotiations, and drop the connection when renegotiations occur.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session negotiate MaxOutstandingR2T=4, and declare MaxRecvDataSegmentLength=512.
- 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 of the session complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection the DUT should start a WRITE operation to transfer at least 4096 bytes.
- When the WRITE command is received the Testing Station should transmit 4 consecutive R2T's on the same connection as the command was received for 512 bytes each.
- The Testing Station should transmit one more R2T 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.
Test #14.4 MaxOutstandingR2T

**Purpose:** To verify that an iSCSI initiator 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:** Mon Jun 30 08:42:34 2003

**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. Initiators are required to detect renegotiations, and drop the connection when renegotiations occur.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal"
- On the leading login of the session negotiate MaxOutstandingR2T=4, and declare MaxRecvDataSegmentLength=512.
- 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 of the session complete the Login Phase and proceed to Full Feature Phase operation.
- On each connection the initiator should start a WRITE operation to transfer at least 4096 bytes.
- On each connection when the WRITE command is received the Testing Station should transmit 4 consecutive R2T’s on the same connection. for 512 bytes each.
- The Testing Station should transmit 4 more R2Ts’ on each connection until R2T’s for all 4096 bytes have been transmitted for each command.

**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.
Test #15.1 DataPDUInOrder

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

**Reference:** 12.18

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

**Last Modification:** Tue Feb 11 08:24:33 2003

**Discussion:** No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. 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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate DataPDUInOrder=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 complete the Login Phase, and proceed to Full Feature Phase.
- On each connection allow the DUT to perform a WRITE 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 #15.2 DataPDUInOrder

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

**Reference:** 12.18

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

**Last Modification:** Thu Feb 20 09:35:56 2003

**Discussion:** No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. 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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate DataPDUInOrder=No.
- 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 complete the Login Phase, and proceed to Full Feature Phase.
- On each connection allow the DUT to perform a READ command.
- The Testing Station should transmit Data-In PDUs in response to the READ command without continuously increasing addresses.

**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-In PDUs. The READ 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 #15.3 DataPDUInOrder

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

**Reference:** 5.3, 12.18

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

**Last Modification:** Tue Feb 11 08:24:21 2003

**Discussion:** No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. 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. Initiators are required to detect the renegotiation of keys. If a renegotiation is detected, the Initiator must drop the connection.

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

**Procedure:**
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session, negotiate DataPDUInOrder=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 DataPDUInOrder=Yes.

**Observable Results:**
· Verify the DUT detects the renegotiation and drops the connection.

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #15.4 DataPDUInOrder

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

**Reference:** 5.3, 12.18

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

**Last Modification:** Mon Feb 3 09:13:57 2003

**Discussion:** No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. 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. Initiators are required to detect the renegotiation of keys. If a renegotiation is detected, the Initiator must drop the connection.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading connection of the session, negotiate DataPDUInOrder=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 DataPDUInOrder=No.

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

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #15.5 DataPDUInOrder

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

Reference: 5.3, 12.18

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

Last Modification: Tue Feb 11 08:24:48 2003

Discussion: No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. 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. Initiators are required to detect the renegotiation of keys. If a renegotiation is detected, the Initiator must drop the connection.

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

Procedure:
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session, negotiate DataPDUInOrder=No.
· 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 DataPDUInOrder=No.

Observable Results:
· Verify the DUT detects the renegotiation and drops the connection.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #15.6 DataPDUInOrder

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

**Reference:** 5.3, 12.18

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

**Last Modification:** Tue Feb 11 08:25:16 2003

**Discussion:** No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. 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. Initiators are required to detect the renegotiation of keys. If a renegotiation is detected, the Initiator must drop the connection.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate DataPDUInOrder=No.
- 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 DataPDUInOrder=Yes.

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

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #16.1 DataSequenceInOrder

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

**Reference:** 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:** Thu May 1 14:00:21 2003

**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:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate DataSequenceInOrder=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 complete the Login Phase, and proceed to Full Feature Phase.
- On each connection allow the DUT to 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 Data-Out 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.
Test #16.2 DataSequenceInOrder

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

Reference: 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: Thu May 1 14:00:11 2003

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:
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session, negotiate DataSequenceInOrder=No.
· 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 complete the Login Phase, and proceed to Full Feature Phase.
· On each connection allow the DUT to perform a READ command. The Testing Station should transmit Data-In Data Sequences in a random order.

Observable Results:
· Verify the DUT does not offer the DataSequenceInOrder key on the second connection.
· Verify that on each connection the DUT accepts Data-In Data Sequences being transmitted in any order.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #16.3 DataSequenceInOrder

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

**Reference:** 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:** Thu May 1 14:00:03 2003

**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. Initiators are required to detect the renegotiation of keys. If a renegotiation is detected, the Initiator must drop the connection.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate DataSequenceInOrder=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 the Testing Station should offer DataSequenceInOrder=Yes.

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

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #16.4 DataSequenceInOrder

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

Reference: 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: Thu May 1 13:59:52 2003

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. Initiators are required to detect the renegotiation of keys. If a renegotiation is detected, the Initiator must drop the connection.

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

Procedure:
· Allow the DUT to open a session of type "Normal".
· On the leading login of the session, negotiate DataSequenceInOrder=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 the Testing Station should offer DataSequenceInOrder=No.

Observable Results:
· Verify the DUT detects the renegotiation and drops the connection.

Possible Problems: If the device does not support more than 1 connection this item is not testable.
Test #16.5 DataSequenceInOrder

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

**Reference:** 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:** Thu May 1 13:59:41 2003

**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. Initiators are required to detect the renegotiation of keys. If a renegotiation is detected, the Initiator must drop the connection.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate DataSequenceInOrder=No.
- 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 the Testing Station should offer DataSequenceInOrder=No.

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

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #16.6 DataSequenceInOrder

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

**Reference:** 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:** Thu May 1 13:59:28 2003

**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. Initiators are required to detect the renegotiation of keys. If a renegotiation is detected, the Initiator must drop the connection.

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

**Procedure:**
- Allow the DUT to open a session of type "Normal".
- On the leading login of the session, negotiate DataSequenceInOrder=No.
- 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 the Testing Station should offer DataSequenceInOrder=Yes.

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

**Possible Problems:** If the device does not support more than 1 connection this item is not testable.
Test #17.1 HeaderDigests and DataDigests

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

**Reference:** 12.1

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

**Last Modification:** Tue Feb 11 08:31:55 2003

**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.
- Allow the DUT to open another connection to the Testing Stations.
- 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 #17.2 HeaderDigests and DataDigests

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

**Reference:** 12.1

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

**Last Modification:** Tue Feb 4 08:07:35 2003

**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.
· Allow the DUT to open another connection to the Testing Stations.
· 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.
Test #18.1 Closed Window

**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:** Tue Feb 11 08:32:39 2003

**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:**
- Allow the DUT to start a connection of SessionType=Normal to the Testing Station. 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 multiple READ operations.
- On each connection the Testing Station should 'close' the command window that the target is willing to receive by sending ExpCmdSN = MaxCmdSN. After a delay of 5 seconds the Testing Station should send a NOP-In PDU with a new ExpCmdSN and MaxCmdSN to indicate an open command window.

**Observable Results:**
- Verify that the DUT starts CmdSN numbering with the first Login Request in the leading connection, and increments CmdSN for every Login Request and Command PDU thereafter.
- Verify that CmdSN is maintained as a Session-wide counter, and does not start back at 0 on the first Login Request of the second connection but is incremented by 1 from the last Login Request or Command PDU sent on the first connection.
- Verify that on both connections that CmdSN is incrementated properly. Verify that no duplicate or out of range CmdSNs are transmitted.
- Verify that on receiving the notification of the closed command window, the DUT does not send anymore SCSI-Command PDUs until the arrival of the NOP-In PDU.
- 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.