iSCSI Consortium
Error Recovery Test Suite
For iSCSI Targets

Version 0.2

Last Update: February 19 2004

iSCSI Consortium
InterOperability Laboratory
Research Computing Center
University of New Hampshire
http://www.iol.unh.edu

121 Technology Drive Suite 2
Durham, NH 03824-3525
Phone: (603) 862-1908
Fax: (603) 862-4181

MODIFICATION RECORD

1. Initial Version 0.1. Version 1.0 is awaiting publication of iSCSI RFC.
2. Version 0.2. Added series of data digest error tests and dropped PDU tests.

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

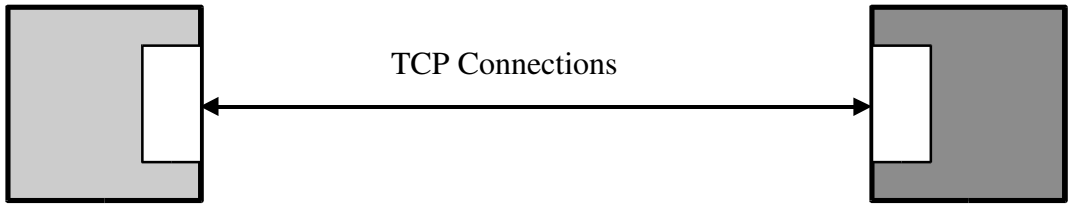
The following documents are referenced in this text:

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

Test #1.1: Retry Inadvertent

Purpose: To see that the DUT properly handles a command that was retried inadvertently.

Reference: 3.2.2.1, 6.2.1

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

Last Modification: Mon Jun 16 13:32:17 2003

Discussion: If initiators, as part of plugging command sequence gaps as described above, inadvertently issue retries for allegiant commands already in progress (i.e., targets did not see the discontinuities in CmdSN ordering), the duplicate commands are silently ignored by targets as specified in section 3.2.2.1.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login and move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· Transmit a WRITE command with the same CmdSN as the TEST_UNIT_READY.
· Transmit a second WRITE command with a CmdSN of one more than that of the TEST_UNIT_READY.

Observable Results:
· Verify that the DUT silently ignores the duplicate CmdSN, and does not send Reject, R2T, or close the connection.
· Verify that the DUT completes the second WRITE command.

Possible Problems: None.

Test #1.2: Retry after Digest Error

Purpose: To see that the DUT properly handles a command that is being retried after it was discarded due to a Digest Error.

Reference: 6.2.1, 6.3, 6.7

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

Last Modification: Mon Jun 16 14:56:03 2003

Discussion: By resending the same iSCSI command PDU ("retry") in the absence of a command acknowledgement (by way of an ExpCmdSN update) or a response, an initiator attempts to "plug" (what it thinks are) the discontinuities in CmdSN ordering on the target end. Discarded command PDUs, due to digest errors, may have created these discontinuities. When a target receives any iSCSI PDU with a payload digest error, it MUST answer with a Reject PDU with a reason code of Data-Digest- Error and discard the PDU. The CmdSN of the rejected command PDU (if it is a non-immediate command) MUST NOT be considered received by the target (i.e., a command sequence gap must be assumed for the CmdSN), even though the CmdSN of the rejected command PDU may be reliably ascertained.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=Yes, DataDigest=CRC32C.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· Transmit a WRITE command with a Data Digest error. Once Reject is received from the DUT the Testing Station should retry the original WRITE command. This is accomplished by sending a WRITE command with identical Initiator Task Tag, flags, function names, LUN, CDB, and CmdSN.

Observable Results:

· Verify that the DUT transmits Reject to the WRITE command received with the Data Digest error.
· Verify that the DUT completes the second WRITE command.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #2.1: Allegiance Reassignment

Purpose: To see that the DUT properly handles a task that is being reassigned after it has had its connection allegiance changed.

Reference: 6.2.2, 10.6, 10.14

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

Last Modification: Tue Jun 17 09:17:21 2003

Discussion: By issuing a "task reassign" task management request (Section 10.5.1 Function), the initiator signals its intent to continue an already active command (but with no current connection allegiance) as part of connection recovery. This means that a new connection allegiance is requested for the command, which seeks to associate it to the connection on which the task management request is being issued. Before the allegiance reassignment is attempted for a task, an implicit or explicit Logout with the reason code "remove the connection for recovery" (see section 10.14) MUST be successfully completed for the previous connection to which the task was allegiant.

Test Setup: The DUT and Test Station pair should be able to make 2 TCP connections.

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, DataDigest=CRC32C.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· Transmit a WRITE command.
· The DUT should transmit the first 2 Data-Out PDU's of the command when R2T's are received. The third Data-Out PDU should be transmitted with a Data Digest error. Once Reject is received from the DUT the Testing Station should open a second connection within the session.
· The Testing Station should transmit a Logout Request with the reason code "remove the connection for recovery" on the first connection.
· On the second connection, after the Login Phase is complete, the Testing Station should transmit a Task Management Request with function code TASK REASSIGN (8). The

DUT is expected to respond with a Task Management Request with response code 0, Function complete. <BR.
· Retransmit the Data-Out PDU that was rejected previously, this time with no Data-Digest error. Transmit all remaining Data-Out PDU's for the command.

Observable Results:
· Verify that the DUT transmits SCSI Response of status Good.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable. The target may indicate that it does not support allegiance reassignment either by transmitting a Logout Response of reason code "connection recovery not supported" (2) on the first connection, or by transmitting a Task Management Response of reason code "Allegiance reassignment not supported" (4) on the second connection. In either of these cases the item is not testable.

Test #2.2: Allegiance Reassignment

Purpose: To see that the DUT properly handles a command that is being reassigned before its connection allegiance has been changed.

Reference: 6.2.2,

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

Last Modification: Tue Jun 17 09:20:31 2003

Discussion: By issuing a "task reassign" task management request (Section 10.5.1 Function), the initiator signals its intent to continue an already active command (but with no current connection allegiance) as part of connection recovery. This means that a new connection allegiance is requested for the command, which seeks to associate it to the connection on which the task management request is being issued. Before the allegiance reassignment is attempted for a task, an implicit or explicit Logout with the reason code "remove the connection for recovery" (see section 10.14) MUST be successfully completed for the previous connection to which the task was allegiant.If allegiance reassignment is supported by the target, but the task is still allegiant to a different connection, or a successful recovery Logout of the previously allegiant connection was not performed, the target MUST respond with a Task Management response code of "Task still allegiant".

Test Setup: The DUT and Test Station pair should be able to make 2 TCP connections.

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, DataDigest=CRC32C.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· Transmit a WRITE command.
· The DUT should transmit the first 2 Data-Out PDU's of the command when R2T's are received. The third Data-Out PDU should be transmitted with a Data Digest error. Once Reject is received from the DUT the Testing Station should open a second connection within the session.

· On the second connection, after the Login Phase is complete, the Testing Station should transmit a Task Management Request with function code TASK REASSIGN (8).

Observable Results:
· Verify that the DUT transmits Task Management Response with response code 3, Task still allegiant.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #3.1: R2T SNACK Support

Purpose: To see that the DUT properly handles SNACK type 0.

Reference: 6.2.2, 10.16

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

Last Modification: Tue Jun 17 11:31:56 2003

Discussion: Any PDU retransmission requests for a currently allegiant command in progress may be made using the SNACK mechanism described in section 10.16, although the usage of SNACK is OPTIONAL.If the implementation supports ErrorRecoveryLevel greater than zero, it MUST support all SNACK types.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should Transmit a WRITE command.
· The Testing Station should transmit the first 2 Data-Out PDU's of the command when R2T's are received.
· Upon receiving the 3rd R2T the Testing Station should transmit a SNACK of type 0 with begRun and RunLength indicating the 3rd received R2T.
· When the DUT retransmits the requested R2T complete the write command by transmitting the final Data-Out PDU with the F bit set to 1.

Observable Results:
· Verify that the DUT retransmits the requested R2T.
· Verify that the DUT transmits a SCSI Response of Status Good once the command is complete.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not

testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #3.2: Data SNACK Support

Purpose: To see that the DUT properly handles SNACK type 0.

Reference: 6.2.2, 10.16

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

Last Modification: Tue Jun 17 11:35:37 2003

Discussion: Any PDU retransmission requests for a currently allegiant command in progress may be made using the SNACK mechanism described in section 10.16, although the usage of SNACK is OPTIONAL.If the implementation supports ErrorRecoveryLevel greater than zero, it MUST support all SNACK types.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, MaxRecvDataSegmentLength=512.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· Transmit a READ command for 2048 bytes.
· The DUT should transmit 4 Data-IN PDU's.
· Upon receiving the 3rd R2T the Testing Station should transmit a SNACK of type 0 with BegRun and RunLength indicating the 3rd received Data-In PDU.

Observable Results:
· Verify that the DUT retransmits the requested Data-In.
· Verify that the DUT transmits a SCSI Response of Status Good once the command is complete.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #3.3: Data SNACK Support

Purpose: To see that the DUT properly handles SNACK type 0 when a run of PDU's is requested.

Reference: 6.2.2, 10.16

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

Last Modification: Tue Jun 17 11:35:31 2003

Discussion: Any PDU retransmission requests for a currently allegiant command in progress may be made using the SNACK mechanism described in section 10.16, although the usage of SNACK is OPTIONAL.If the implementation supports ErrorRecoveryLevel greater than zero, it MUST support all SNACK types.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, MaxRecvDataSegmentLength=512.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· Transmit a READ command for 2048 bytes.
· The DUT should transmit 4 Data-IN PDU's.
· Upon receiving the 3rd R2T the Testing Station should transmit a SNACK of type 0 with BegRun and RunLength indicating all 3 received Data-In PDUs.

Observable Results:
· Verify that the DUT retransmits the requested Data-In PDUs.
· Verify that the DUT transmits a SCSI Response of Status Good once the command is complete.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the

connection, this item is not testable.

Test #3.4: Data SNACK Support

Purpose: To see that the DUT properly handles SNACK type 0 when a run of PDU's is requested.

Reference: 6.2.2, 10.16

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

Last Modification: Tue Jun 17 11:38:24 2003

Discussion: Any PDU retransmission requests for a currently allegiant command in progress may be made using the SNACK mechanism described in section 10.16, although the usage of SNACK is OPTIONAL.If the implementation supports ErrorRecoveryLevel greater than zero, it MUST support all SNACK types.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, MaxRecvDataSegmentLength=512.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· Transmit a READ command for 2048 bytes.
· The DUT should transmit 4 Data-IN PDU's.
· Upon receiving the 4th R2T the Testing Station should transmit a SNACK of type 0 with BegRun and RunLength set to 0.

Observable Results:
· Verify that the DUT retransmits all 4 of the requested Data-In PDUs.
· Verify that the DUT transmits a SCSI Response of Status Good once the command is complete.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the

connection, this item is not testable.

Test #3.5: Status SNACK Support

Purpose: To see that the DUT properly handles a SNACK type 1 request.

Reference: 6.2.2, 10.16.1

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

Last Modification: Tue Jun 17 12:24:54 2003

Discussion: Any PDU retransmission requests for a currently allegiant command in progress may be made using the SNACK mechanism described in section 10.16, although the usage of SNACK is OPTIONAL.If the implementation supports ErrorRecoveryLevel greater than zero, it MUST support all SNACK types.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, MaxRecvDataSegmentLength=512.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· Perform 3 consecutive WRITE commands.
· Once status has been received for all 3 WRITE commands, transmit a SNACK of type 1 with BegRun and RunLength indicating status is requested for the last received Status.

Observable Results:
· Verify that the DUT retransmits the requested Response PDU.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #3.6: Status SNACK Support

Purpose: To see that the DUT properly handles a SNACK type 1 request for a run of numbered responses.

Reference: 6.2.2, 10.16.1

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

Last Modification: Tue Jun 17 12:24:35 2003

Discussion: Any PDU retransmission requests for a currently allegiant command in progress may be made using the SNACK mechanism described in section 10.16, although the usage of SNACK is OPTIONAL.If the implementation supports ErrorRecoveryLevel greater than zero, it MUST support all SNACK types.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, MaxRecvDataSegmentLength=512.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· Perform 3 consecutive WRITE commands.
· Once status has been received for all 3 WRITE commands, transmit a SNACK of type 1 with BegRun and RunLength both of 0.

Observable Results:
· Verify that the DUT retransmits all 4 of the requested Response PDUs.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #3.7: Resegmentation SNACK Support

Purpose: To see that the DUT properly handles a SNACK type 3 request when resegmentation is necessary.

Reference: 6.2.2, 10.16.1, 10.16.3

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

Last Modification: Tue Jun 17 12:37:33 2003

Discussion: With R-Data SNACK, the initiator indicates that it discards all the unacknowledged data and expects the target to resend it. It also expects resegmentation. In this case, the retransmitted Data-In PDUs MAY be different from the ones originally sent in order to reflect changes in MaxRecvDataSegmentLength. Their DataSN starts with the BegRun of the last DataACK received by the target if any was received; otherwise it starts with 0 and is increased by 1 for each resent Data-In PDU. A target that has received a R-Data SNACK MUST return a SCSI Response that contains a copy of the SNACK Tag field from the R-Data SNACK in the SCSI Response SNACK Tag field as its last or only Response. For example, if it has already sent a response containing another value in the SNACK Tag field or had the status included in the last Data-In PDU, it must send a new SCSI Response PDU. If a target sends more than one SCSI Response PDU due to this rule, all SCSI responses must carry the same StatSN (see Section 10.4.4 SNACK Tag). If an initiator attempts to recover a lost SCSI Response (with a Status-SNACK, see Section 10.16.1 Type) when more than one response has been sent, the target will send the SCSI Response with the latest content known to the target, including the last SNACK Tag for the command.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, MaxRecvDataSegmentLength=1024.
· The Testing Station should move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.

· The Testing Station should transmit a READ command for 4096 bytes.
· Once status has been received for the READ command, the Testing Station should send a Text Request declaring a new value of MaxRecvDataSegmentLength=512. Wait for a valid Text Response.
The Testing Station should transmit a SNACK of type 3 with BegRun and RunLength both of 0.

Observable Results:
· Verify that the DUT retransmits all of the Data for the READ command using the new MaxRecvDataSegment Length.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #4.1: Usage of Reject Non-Command PDU

Purpose: To see that the if the DUT chooses to issue a Reject PDU for an errored Non-Command PDU, it also transmits SCSI Response when the command is complete.

Reference: 6.3, 6.7

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

Last Modification: Thu Jul 24 11:06:42 2003

Discussion: Targets MUST NOT implicitly terminate an active task by sending a Reject PDU for any PDU exchanged during the life of the task. If the target decides to terminate the task, a Response PDU (SCSI, Text, Task, etc.) must be returned by the target to conclude the task.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command for 4096 bytes.
· Once R2T is received the Testing Station shold transmit a Data-Out PDU with a payload-digest error.

Observable Results:
· Verify that the DUT transmits a Reject PDU, and that it also transmits a SCSI Response PDU when the command is complete, and not before a Data-Out with the F bit set is received.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #4.2.1: Usage of Reject Command PDU

Purpose: To see that the if the DUT chooses to issue a Reject PDU for an errored Command PDU, it does not transmit any response.

Reference: 6.3

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

Last Modification: Thu Jul 24 15:32:11 2003

Discussion: Targets MUST NOT implicitly terminate an active task by sending a Reject PDU for any PDU exchanged during the life of the task. If the target decides to terminate the task, a Response PDU (SCSI, Text, Task, etc.) must be returned by the target to conclude the task.The CmdSN of the rejected command PDU (if it is a non-immediate command) MUST NOT be considered received by the target (i.e., a command sequence gap must be assumed for the CmdSN), even though the CmdSN of the rejected command PDU may be reliably ascertained.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=Yes.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command for 4096 bytes with ImmediateData attached. There should be a data-digest error.
· Once the Reject PDU is received transmit a second WRITE command with the same CmdSN.

Observable Results:
· Verify that the DUT transmits a Reject PDU. Verify that the DUT does not transmits a SCSI Response PDU or any other response such as R2T.
· Verify that the DUT responds to the second received WRITE command properly, not ignoring it as if it were a duplicate CmdSN.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #4.2.2: Usage of Reject Command PDU

Purpose: To see that the if the DUT chooses to issue a Reject PDU for an errored Command PDU, it does not transmit any response.

Reference: 6.3

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

Last Modification: Thu Jul 24 11:05:43 2003

Discussion: Targets MUST NOT implicitly terminate an active task by sending a Reject PDU for any PDU exchanged during the life of the task. If the target decides to terminate the task, a Response PDU (SCSI, Text, Task, etc.) must be returned by the target to conclude the task.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=Yes.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a READ command for 4096 bytes. The R bit should not be set.

Observable Results:
· Verify that if the DUT transmits a Reject PDU, it also does not transmits a SCSI Response PDU.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #5.1: Termination of Tasks

Purpose: To see that the if the DUT terminates tasks under the necessary conditions.

Reference: 6.5

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

Last Modification: Thu Jul 24 15:42:39 2003

Discussion: A target implicitly terminates the active tasks due to iSCSI protocol dynamics when a connection is implicitly or explicitly logged out with the reason code of "Close the connection" and there are active tasks allegiant to that connection.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes.
· The Testing Station should move into the Full Feature Phase. Open a second connection to the DUT and move into Full Feature Phase.
· On each connection the Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command for 4096 bytes with ImmediateData attached.
· Once the R2T PDU is received, the Testing Station should transmit a Logout Request PDU with reason code of 'Close the Connection'.
· On the second connection transmit a Task Management Request to the DUT of Function Code 8 TASK REASSIGN.

Observable Results:
· Verify that the DUT transmits a proper Logout Response.
· Verfify that the DUT does not attempt to fulfill the Task Management Request, but rather responds with a Task Management Response indicating that the task could not be reassigned, shuch as Response Code 1 Task does not exist.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #5.2: Termination of Tasks

Purpose: To see that the if the DUT terminates tasks under the necessary conditions.

Reference: 6.5

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

Last Modification: Thu Jul 24 15:46:01 2003

Discussion: A target implicitly terminates the active tasks due to iSCSI protocol dynamics when a connection fails and eventually the connection state times out and there are active tasks allegiant to that connection.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes.
· The Testing Station should move into the Full Feature Phase. Open a second connection to the DUT and move into Full Feature Phase.
· On each connection the Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command for 4096 bytes with ImmediateData attached.
· Once the R2T PDU is received, the Testing Station should drop the connection.
· On the second connection wait 2 minutes and then transmit a Task Management Request to the DUT of Function Code 8 TASK REASSIGN.

Observable Results:
· Verfify that the DUT does not attempt to fulfill the Task Management Request, but rather responds with a Task Management Response indicating that the task could not be reassigned, shuch as Response Code 1 Task does not exist.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not

testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #5.3: Termination of Tasks

Purpose: To see that the if the DUT terminates tasks under the necessary conditions.

Reference: 6.5

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

Last Modification: Thu Jul 24 15:49:29 2003

Discussion: A target implicitly terminates the active tasks due to iSCSI protocol dynamics when a successful Logout with the reason code of "remove the connection for recovery" is performed while there are active tasks allegiant to that connection, and those tasks eventually time out after the Time2Wait and Time2Retain periods without allegiance reassignment.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes, DefaultTime2Wait=2, DefaultTime2Retain=5.
· The Testing Station should move into the Full Feature Phase. Open a second connection to the DUT and move into Full Feature Phase.
· On each connection the Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command for 4096 bytes with ImmediateData attached.
· Once the R2T PDU is received, the Testing Station should transmit a Logout Request with reason code "Remove Connection for Recovery'.
· On the second connection wait 8 seconds and then transmit a Task Management Request to the DUT of Function Code 8 TASK REASSIGN.

Observable Results:
· Verfify that the DUT does not attempt to fulfill the Task Management Request, but rather responds with a Task Management Response indicating that the task could not be

reassigned, shuch as Response Code 1 Task does not exist.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #5.4: Termination of Tasks

Purpose: To see that the if the DUT terminates tasks under the necessary conditions.

Reference: 6.5

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

Last Modification: Mon Aug 4 15:00:12 2003

Discussion: A target implicitly terminates the active tasks due to iSCSI protocol dynamics when a connection is implicitly or explicitly logged out with the reason code of "Close the session" and there are active tasks in that session.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes, DefaultTime2Wait=2, DefaultTime2Retain=5.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command for 4096 bytes with ImmediateData attached.
· Once the R2T PDU is received, the Testing Station should transmit a Logout Request with reason code 'Close the Session'.
· Once the session is closed the Testing Station should open a new session. During the Login Phase negotiate ErrorRecoveryLevel>0. Once in the Full Feature Phase the Testing Station should transmit a Task Management Request to the DUT of Function Code 8 TASK REASSIGN.

Observable Results:
· Verfify that the DUT does not attempt to fulfill the Task Management Request, but rather responds with a Task Management Response indicating that the task could not be reassigned, shuch as Response Code 1 Task does not exist.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #6.1: Format Error

Purpose: To see that the if the DUT properly identifies and reacts to a format error.

Reference: 6.6

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

Last Modification: Mon Aug 4 14:59:32 2003

Discussion: The following two explicit violations of PDU layout rules are format errors: a) Illegal contents of any PDU header field except the Opcode (legal values are specified in Section 10 iSCSI PDU Formats)b) Inconsistent field contents (consistent field contents are specified in Section 10 iSCSI PDU Formats). Format errors indicate a major implementation flaw in one of the parties. When a target or an initiator receives an iSCSI PDU with a format error, it MUST immediately terminate all transport connections in the session either with a connection close or with a connection reset and escalate the format error to session recovery.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes, DefaultTime2Wait=2, DefaultTime2Retain=5.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command with the AHSlength field set to a non zero value, but no AHS field in the PDU.

Observable Results:
· Verfify that the DUT does not attempt to fulfill the Task Management Request, but rather responds with a Task Management Response indicating that the task could not be reassigned, shuch as Response Code 1 Task does not exist.

Possible Problems: If the target does not support ErrorRecoveryLevel>0 this item is not

testable. If the target chooses to escalate this error to Session Recovery by closing the connection, this item is not testable.

Test #7.1: Header Digest Error

Purpose: To see that the if the DUT properly identifies and reacts to a received header digest error.

Reference: 6.7

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

Last Modification: Tue Aug 5 09:04:08 2003

Discussion: When a target or an initiator receives any iSCSI PDU, with a header digest error, it MUST either discard the header and all data up to the beginning of a later PDU or close the connection.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=Yes, InitialR2T=No.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command with immediate data attached, the F bit set to 1, and a header digest error.
· Transmit a second valid WRITE command with a new CmdSN and a new InitiatorTaskTag.

Observable Results:
· Verfify that the DUT either ignores the first received WRITE command, and responds only to the second received WRITE command, or closes the connection and starts Session Recovery.

Possible Problems: None.

Test #7.2: Data Digest Error

Purpose: To see that the if the DUT properly identifies and reacts to a received payload digest error.

Reference: 6.7

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

Last Modification: Tue Aug 5 09:03:48 2003

Discussion: When a target receives any iSCSI PDU with a payload digest error, it MUST answer with a Reject PDU with a reason code of Data-Digest-Error and discard the PDU.No further action is necessary for targets if the discarded PDU is a non-data PDU. In case of immediate data being present on a discarded command, the immediate data is implicitly recovered when the task is retried followed by the entire data transfer for the task.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=Yes, InitialR2T=No.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command with immediate data attached, the F bit set to 1, and a data digest error.
· Transmit a second valid WRITE command with a new CmdSN and a new InitiatorTaskTag.

Observable Results:
· Verfify that the DUT transmit a Reject PDU with reason code of Data-Digest-Error, and then takes no further action to recover the PDU.
· Verify that the DUT responds to the second received WRITE command, or closes the connection and starts Session Recovery.

Possible Problems: None.

Test #7.3: Data Digest Error

Purpose: To see that the if the DUT properly identifies and reacts to a received data digest error.

Reference: 6.7

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

Last Modification: Tue Aug 5 09:08:00 2003

Discussion: When a target receives any iSCSI PDU with a payload digest error, it MUST answer with a Reject PDU with a reason code of Data-Digest-Error and discard the PDU.If the discarded PDU is a solicited or unsolicited iSCSI data PDU the target MUST do one of the following a) Request retransmission with a recovery R2T. b) Terminate the task with a response PDU with a CHECK CONDITION Status and an iSCSI Condition of "protocol service CRC error". If the target chooses to implement this option, it MUST wait to receive all the data (signaled by a Data PDU with the final bit set for all outstanding R2Ts) before sending the response PDU. A task management command (such as an abort task) from the initiator during this wait may also conclude the task.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes, DefaultTime2Wait=2, DefaultTime2Retain=5.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command.
· Once R2T is received from the DUT the Testing Station should start sending Data-Out PDUs to satisfy the R2T. One of the Data-Out PDUs sent should have a Data-Digest-Error. This should not be the final PDU in the sequence (i.e. does not have the F bit set).
· Transmit a second valid WRITE command with a new CmdSN and a new InitiatorTaskTag.

Observable Results:
· Verfify that the DUT transmit a Reject PDU with reason code of Data-Digest-Error, and then takes no further action to recover the PDU.
· Verify that the DUT either transmits a Recovery R2T or sends status of CHECK CONDITION. If the DUT chooses to send status of CHECK CONDITION it must wait until the final Data-Out PDU is received.

Possible Problems: None.

Test #8.1: Out of Order DataSN

Purpose: To see that the if the DUT properly identifies and reacts to a PDU with an out of order DataSN.

Reference: 6.8

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

Last Modification: Tue Aug 5 09:42:41 2003

Discussion: When a target receives a data PDU with an out of order DataSN, it means that the target must have hit a header or payload digest error on at least one of the earlier data PDUs. The target MUST address these implied digest errors as described in Section 6.7 Digest Errors.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes, DefaultTime2Wait=2, DefaultTime2Retain=5.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command.
· Once R2T is received from the DUT the Testing Station should start sending Data-Out PDUs to satisfy the R2T. One of the Data-Out PDUs sent should have a DataSn of 2 greater than the last Data-Out, as opposed to 1 greater as would normally be the case. This should not be the final PDU in the sequence (i.e. does not have the F bit set). This will have the effect of 'skipping' one DataSN. Data-PDU's should be transmittmed until the R2T is satisfied, continuing the DataSN count from the new DataSN.

Observable Results:
· Verfify that the DUT does one of the following: 1) Close the Connection or 2) Transmit Reject PDU. If the DUT chooses 2 it MUST either then request retransmission with recovery R2T or transmit a SCSI Response with status CHECK CONDITION.

Possible Problems: None.

Test #8.2: Out of Order DataSN

Purpose: To see that the if the DUT properly identifies and reacts to a PDU with an out of order DataSN.

Reference: 6.8

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

Last Modification: Tue Aug 5 09:43:52 2003

Discussion: When a target receives a data PDU with an out of order DataSN, it means that the target must have hit a header or payload digest error on at least one of the earlier data PDUs. The target MUST address these implied digest errors as described in Section 6.7 Digest Errors.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes, DefaultTime2Wait=2, DefaultTime2Retain=5.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command.
· Once R2T is received from the DUT the Testing Station should start sending Data-Out PDUs to satisfy the R2T. One of the Data-Out PDUs sent should have a DataSn of 2 greater than the last Data-Out, as opposed to 1 greater as would normally be the case. This should not be the final PDU in the sequence (i.e. does not have the F bit set). Data-PDU's should be transmittmed until the R2T is satisfied, continuing the DataSN count from the original DataSN. This will have the effect of 'swapping' two DataSN values.

Observable Results:
· Verfify that the DUT does one of the following: 1) Close the Connection or 2) Transmit Reject PDU. If the DUT chooses 2 it MUST either then request retransmission with recovery R2T or transmit a SCSI Response with status CHECK CONDITION.

Possible Problems: None.

Test #8.3: Out of Order DataSN

Purpose: To see that the if the DUT properly identifies and reacts to a PDU with an out of order DataSN.

Reference: 6.8

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

Last Modification: Tue Aug 5 09:46:26 2003

Discussion: When a target receives a data PDU with an out of order DataSN, it means that the target must have hit a header or payload digest error on at least one of the earlier data PDUs. The target MUST address these implied digest errors as described in Section 6.7 Digest Errors.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes, DefaultTime2Wait=2, DefaultTime2Retain=5.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command.
· Once R2T is received from the DUT the Testing Station should start sending Data-Out PDUs to satisfy the R2T. One of the Data-Out PDUs sent should have a DataSN greater than the current DataSN by more than the number of Data PDUs in the sequence. This should not be the final PDU in the sequence (i.e. does not have the F bit set). Data-PDU's should be transmittmed until the R2T is satisfied, continuing the DataSN count from the original DataSN.

Observable Results:
· Verfify that the DUT does one of the following: 1) Close the Connection or 2) Transmit Reject PDU. If the DUT chooses 2 it MUST either then request retransmission with recovery R2T or transmit a SCSI Response with status CHECK CONDITION.

Possible Problems: None.

Test #9.1: Protocol Error

Purpose: To see that the if the DUT properly identifies and reacts to a PDU with an out of order DataSN.

Reference: 6.8

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

Last Modification: Tue Aug 5 10:56:54 2003

Discussion: When a target receives a data PDU with an out of order DataSN, it means that the target must have hit a header or payload digest error on at least one of the earlier data PDUs. The target MUST address these implied digest errors as described in Section 6.7 Digest Errors.

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

Procedure:
· Open a normal session to the DUT.
· Negotiate the following keys: ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes, DefaultTime2Wait=2, DefaultTime2Retain=5.
· The Testing Station should move into the Full Feature Phase.
· The Testing Station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command.
· Once R2T is received from the DUT the Testing Station should start sending Data-Out PDUs to satisfy the R2T. One of the Data-Out PDUs sent should have a DataSN greater than the current DataSN by more than the number of Data PDUs in the sequence. This should not be the final PDU in the sequence (i.e. does not have the F bit set). Data-PDU's should be transmittmed until the R2T is satisfied, continuing the DataSN count from the original DataSN.

Observable Results:
· Verfify that the DUT does one of the following: 1) Close the Connection or 2) Transmit Reject PDU. If the DUT chooses 2 it MUST either then request retransmission with recovery R2T or transmit a SCSI Response with status CHECK CONDITION.

Possible Problems: None.

Test #10.1: Drop Immediate Command

Purpose: To see that the DUT properly handles the recovery actions of an Initiator when an immediate command has been dropped.

Reference: 6.1.4.2, 6.2

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

Last Modification: Thu Feb 12 12:27:18 2004

Discussion: A request not acknowledged for a long time is an error which lends itself to within connection recovery. Requests are acknowledged explicitly through ExpCmdSN. An initiator may retry non-acknowledged commands.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ErrorRecoveryLevel>0 and move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should simulate an immediate READ command being dropped in transit to the DUT.
· Transmit a second READ command with the same CmdSN as the dropped command PDU, although this time without the immediate bit set.
· The Testing Station should retry the dropped PDU, transmitting a READ command with the immediate bit set and the same CmdSN as the previous READ command.

Observable Results:
· Verify that the DUT properly responds to both received commands and does not transmit Reject, close the connection, or ignore the duplicate CmdSN.

Possible Problems: None.

Test #10.2: Drop Non-Immediate Command

Purpose: To see that the DUT properly handles the recovery actions of an Initiator when a command has been dropped.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 10 15:04:51 2004

Discussion: A request not acknowledged for a long time is an error which lends itself to within connection recovery. Requests are acknowledged explicitly through ExpCmdSN. An initiator may retry non-acknowledged commands.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ErrorRecoveryLevel>0 and move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should simulate a READ command being dropped in transit to the DUT.
· Transmit a new READ command with the CmdSN one more than the previous dropped command.
· The Testing Station should retry the dropped PDU, transmitting a READ command with the same CmdSN as the original READ command.

Observable Results:
· Verify that the DUT properly responds to both received commands and does not transmit Reject, close the connection, or ignore either command.

Possible Problems: None.

Test #11.1: Drop Solicited Data-Out

Purpose: To see that the DUT properly handles the recovery actions when a solicited Data-Out PDU has been dropped.

Reference: 6.1.4.1, 6.7, 6.8

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

Last Modification: Tue Feb 10 16:22:42 2004

Discussion: At the target when an Data-Out PDU is received with an out of order DataSN, it implies that the target encountered a digest error on a previous Data-Out PDU. The target MUST address these implied digest errors as it would a detected digest error. The target has several options. It may use the option of a recovery R2T. It may terminate the task with CHECK CONDITION after a PDU with the F bit set is recieved. The target finally may chose to close the connection.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ErrorRecoveryLevel>0, ImmediateData=No, InitialR2T=Yes,MaxOutstandingR2Ts=4 MaxBurstSize=4096, MaxRecvDataSegmentLength=1024, and move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a WRITE command for 4096 bytes to the DUT, then wait for R2Ts from the DUT for all data associated with the command.
· The Testing Station should transmit 4 Data-Out PDUs to the DUT, with the second one being dropped before it reaches the DUT.

Observable Results:
· Verify that the DUT either transmits a Recovery R2T, terminates the task with CHECK CONDITION once the Data-Out with F=1 is received, or closes the connection.

Possible Problems: None.

Test #12.1: Drop Data-In

Purpose: To see that the DUT properly handles the recovery actions of an Initiator when a Data-In PDU has been dropped.

Reference: 6.1.4.1, 6.8

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

Last Modification: Tue Feb 10 16:20:23 2004

Discussion: At the Initiator a lost Data-In PDU error lends itself to within-command recovery. This will be detected usually by a sequence reception timeout. The initiator can handle this error using SNACK.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ErrorRecoveryLevel>0,MaxRecvDataSegmentLength=1024, and move into the Full Feature Phase.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a READ command for 4096 bytes to the DUT, then wait for 4 Data-In PDUs to the DUT. The Testing Station should behave as if the second Data-In PDU was dropped and transmit SNACK to request the second Data-In PDU from the DUT.

Observable Results:
· Verify that the DUT either retransmits the dropped Data-In PDU or transmits SNACK reject. If the DUT transmits SNACK reject and status has not already been sent for the command, the target must terminate the command with status CHECK CONDITION.

Possible Problems: None.

Test #13.1: Drop Text Response

Purpose: To see that the DUT properly handles the recovery actions of an Initiatior when a Text Response PDU has been dropped.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 08:30:38 2004

Discussion: At the Initiator a lost iSCSI numbered response may be detected as a sequence error. This error can be recovered using the SNACK option.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a Text Request to the DUT to with a vendor specifc key=value pair in the extedned key format. The DUT is expected to respond with a Text Response, however the Testing Station will simulate this response PDU being dropped.
·After 1 second, the Testing Station should retry the Text Request, sending an identical copy of the original Text Request. The DUT is expected to ignore this retry.
·After 1 more second, the Testing Station should retry the Text Request a second time, sending an identical copy of the original Text Request. The DUT is expected to ignore this retry.
· The Testing Station should transmit a NOP-Out ping. The DUT is expected to respond with a NOP-In repsonse.
· The Testing Station should transmit a SNACK to request retransmission of the dropped Text Response PDU.

Observable Results:
· Verify that the DUT silently ignores any command retries from the Testing Station.

· Verify that the DUT retransmits the PDU when SNACK is received. The Target may also transmit SNACK Reject followed by CHECK CONDITION.
· Verify that the NOP-In transmitted by the DUT carries an appropriate StatSN.

Possible Problems: None.

Test #13.2: Drop Text Request

Purpose: To see that the DUT properly handles the recovery actions of an Initiatior when a Text Request PDU has been dropped.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 08:30:56 2004

Discussion: At the Initiator a request not acknowledged for a long time, may lend itself to within connection recovery. An initiator may retry non-acknowledged commands.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a Text Request to the DUT to with a vendor specifc key=value pair in the extended key format, however this PDU will be dropped before it reaches the DUT. The DUT should send a second Text Request to the DUT a different vendor specific key=value pair. The DUT is expected to respond with a Text Response
·After 1 second, the Testing Station should retry the Text Request, sending an identical copy of the original Text Request.

Observable Results:
· Verify that the DUT transmits a Text Response to both Text Requests that are received.

Possible Problems: None.

Test #14.1: Drop NOP-Out

Purpose: To see that the DUT properly handles the recovery actions of an Initiatior when a NOP-Out PDU has been dropped.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 08:31:11 2004

Discussion: At the Initiator a request not acknowledged for a long time, may lend itself to within connection recovery. An initiator may retry non-acknowledged commands.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a NOP-Out to the DUT, however this PDU will be dropped before it reaches the DUT. The DUT should send a second NOP-Out to the DUT. The DUT is expected to respond with a NOP-In.
·After 1 second, the Testing Station should retry the original NOP-Out, sending an identical copy of the original NOP-Out.

Observable Results:
· Verify that the DUT transmits a NOP_in to each NOP-Out received.

Possible Problems: None.

Test #14.2: Drop NOP-In

Purpose: To see that the DUT properly handles the recovery actions of an Initiatior when a NOP-In has been dropped.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 08:31:18 2004

Discussion: At the Initiator a lost iSCSI numbered response may be detected as a sequence error. This error can be recovered using the SNACK option.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a NOP_Out to the DUT. The DUT is expected to reply with a NOP-In response, however the Testing Station will simulate this response PDU being dropped.
·After 1 second, the Testing Station should retry the NOP-Out, sending an identical copy of the original Text Request. The DUT is expected to ignore this retry.
·After 1 more second, the Testing Station should retry the NOP-Out a second time, sending an identical copy of the original NOP-Out. The DUT is expected to ignore this retry.
· The Testing Station should transmit a new NOP-Out ping. The DUT is expected to respond with a NOP-In repsonse.
· The Testing Station should transmit a SNACK to request retransmission of the dropped NOP-In PDU.

Observable Results:
· Verify that the DUT silently ignores any retries from the Testing Station.

· Verify that the DUT retransmits the PDU when SNACK is received. The Target may also transmit SNACK Reject followed by CHECK CONDITION.
· Verify that the NOP-In transmitted by the DUT carries an appropriate StatSN.

Possible Problems: None.

Test #15.1: Data Digest Error on Immediate Data on Immediate Command

Purpose: To see that the DUT properly handle a data digest error on an immediate command with immediate data attached.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 08:31:26 2004

Discussion: At the Target, non-data a PDU with a payload digest error is always answered with a Reject PDU then discarded.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an immediate WRITE command to the DUT to with immediate data attached. This PDU should have a payload digest error. The DUT is expected to respond with a Reject PDU.
·After receiving the Reject PDU the Testing Station should retry the immediate WRITE command, sending an identical copy of the original command.

Observable Results:
· Verify that the DUT transmits Login Reject to the received data digest error.
· Verify that the retried command ends with status GOOD.

Possible Problems: None.

Test #15.2: Data Digest Error on Immediate Data

Purpose: To see that the DUT properly handle a data digest error on a command with immediate data attached.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 08:31:34 2004

Discussion: At the Target, non-data a PDU with a payload digest error is always answered with a Reject PDU then discarded.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an WRITE command to the DUT to with immediate data attached. This PDU should have a payload digest error. The DUT is expected to respond with a Reject PDU.
·After receiving the Reject PDU the Testing Station should retry the immediate WRITE command, sending an identical copy of the original command.

Observable Results:
· Verify that the DUT transmits Login Reject to the received data digest error.
· Verify that the retried command ends with status GOOD.

Possible Problems: None.

Test #16.1: Data Digest Error on Unsolicited Data when F=0

Purpose: To see that the DUT properly handles a data digest error on an Unsolicited data PDU.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 08:56:56 2004

Discussion: At the Target, a PDU with a payload digest error is always answered with a Reject PDU then discarded. The target then has the option of terminating the task once the final PDU of the sequence is received, or transmitting a recovery R2T.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=No. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an WRITE command to the DUT, followed by 4 Data-Out PDUs. The first Data-Out PDU should have a data digest error. The DUT is expected to transmit a Reject PDU.

Observable Results:
· Verify that the DUT either transmits a Recovery R2T, or wait for the final Data-Out PDU and then terminates the task with status CHECK CONDITION.

Possible Problems: None.

Test #16.2: Data Digest Error on Unsolicited Data when F=1

Purpose: To see that the DUT properly handles a data digest error on an Unsolicited data PDU.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 08:56:12 2004

Discussion: At the Target, a PDU with a payload digest error is always answered with a Reject PDU then discarded. If the rejected PDU is the final Data-Out PDU of a command, than the target cannot terminate the task. If the target does not transmit a Recovery R2T, it will need to wait for the initiator to attempt recovery actions.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=No. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an WRITE command to the DUT, followed by 4 Data-Out PDUs. The fourth Data-Out PDU should have a data digest error. The DUT is expected to transmit a Reject PDU.

Observable Results:
· Verify that the DUT transmits a Recovery R2T.

Possible Problems: None.

Test #17.1: Data Digest Error on Solicited Data when F=0

Purpose: To see that the DUT properly handles a data digest error on a solicited data PDU.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 09:21:17 2004

Discussion: At the Target, a PDU with a payload digest error is always answered with a Reject PDU then discarded. The target then has the option of terminating the task once the final PDU of the sequence is received, or transmitting a recovery R2T.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an WRITE command to the DUT, then wait for R2T.
· Once R2T is received from the DUT, the Testing Station should transmit Data-Out PDUs. The first Data-Out PDU should have a data digest error. The DUT is expected to transmit a Reject PDU.

Observable Results:
· Verify that the DUT either transmits a Recovery R2T, or wait for the final Data-Out PDU and then terminates the task with status CHECK CONDITION.

Possible Problems: None.

Test #17.2: Data Digest Error on Solicited Data when F=1

Purpose: To see that the DUT properly handles a data digest error on a solicited data PDU.

Reference: 6.1.4.2, 6.2

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

Last Modification: Tue Feb 17 09:23:03 2004

Discussion: At the Target, a PDU with a payload digest error is always answered with a Reject PDU then discarded. The target then has the option of terminating the task once the final PDU of the sequence is received, or transmitting a recovery R2T.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an WRITE command to the DUT, then wait for R2T.
· Once R2T is received from the DUT, the Testing Station should transmit Data-Out PDUs. The final Data-Out PDU should have a data digest error. The DUT is expected to transmit a Reject PDU.

Observable Results:
· Verify that the DUT transmits a Recovery R2T.

Possible Problems: None.

Test #18.1: Data Digest Error on Data-In when F=0

Purpose: To see that the DUT properly handles the recovery actions of an initiator detecting a data digest error on a Data-In PDU.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 12:33:53 2004

Discussion: At the Initiator, a Data-In PDU with a payload digest error is discarded. The Initiator will then either request retransmission with SNACK or abort the task. The target then has the option of retransmitting the Data-In or transmitting SNACK Reject.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an READ command to the DUT, then wait for Data-In PDUs.
· Once the Testing Station starts receiving Data-Ins, it should simulate a Data Digest Error on a Data-In PDU with F=0. The Testing Station should transmit SNACK for the 'dropped' PDU.

Observable Results:
· Verify that the DUT either retransmits the PDU or transmits SNACK Reject.

Possible Problems: None.

Test #18.2: Data Digest Error on Data-In when F=1

Purpose: To see that the DUT properly handles the recovery actions of an initiator detecting a data digest error on a Data-In PDU.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 12:36:21 2004

Discussion: At the Initiator, a Data-In PDU with a payload digest error is discarded. The Initiator will then either request retransmission with SNACK or abort the task. The target then has the option of retransmitting the Data-In or transmitting SNACK Reject.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an READ command to the DUT, then wait for Data-In PDUs.
· Once the Testing Station starts receiving Data-Ins, it should simulate a Data Digest Error on a Data-In PDU with F=1. The Testing Station should transmit SNACK for the 'dropped' PDU.

Observable Results:
· Verify that the DUT either retransmits the PDU or transmits SNACK Reject.

Possible Problems: None.

Test #19.1: Data Digest Error on NOP-In

Purpose: To see that the DUT properly handles the recovery actions of an initiator detecting a data digest error on a NOP-In PDU.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 12:46:04 2004

Discussion: At the Initiator, any PDU with a payload digest error is discarded. The Initiator will then either request retransmission with SNACK or abort the task. The target then has the option of retransmitting the Data-In or transmitting SNACK Reject.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a NOP-Out ping to the DUT, then wait for NOP-In response.
· Once the Testing Station receives the NOP-In, it should simulate a Data Digest Error on that NOP-In. The Testing Station should transmit SNACK for the 'dropped' PDU.

Observable Results:
· Verify that the DUT either retransmits the PDU or transmits SNACK Reject.

Possible Problems: None.

Test #19.2: Data Digest Error on Immediate NOP-In

Purpose: To see that the DUT properly handles the recovery actions of an initiator detecting a data digest error on a NOP-In PDU.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 12:45:53 2004

Discussion: At the Initiator, any PDU with a payload digest error is discarded. The Initiator will then either request retransmission with SNACK or abort the task. The target then has the option of retransmitting the Data-In or transmitting SNACK Reject.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an immediate NOP-Out ping to the DUT, then wait for NOP-In response.
· Once the Testing Station receives the NOP-In, it should simulate a Data Digest Error on that NOP-In. The Testing Station should transmit SNACK for the 'dropped' PDU.

Observable Results:
· Verify that the DUT either retransmits the PDU or transmits SNACK Reject.

Possible Problems: None.

Test #20.1: Data Digest Error on NOP-Out

Purpose: To see that the DUT properly handles a data digest error on a NOP-Out PDU.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 13:11:42 2004

Discussion: At the target, any non-data PDU with a payload digest error is discarded and a Reject PDU is sent. The Initiator may then retry the PDU.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a NOP-Out ping to the DUT. This NOP-Out should have a Data Digest Error.
· The DUT is expected to transmit a Reject PDU.
· The Testing Station should transmit a READ command to the DUT with a CmdSN of one more than the rejected NOP-Out PDU, followed by a NOP-Out PDU with the same CmdSN as the original.

Observable Results:
· Verify that the DUT transmits a Reject PDU to the first NOP-Out PDU. Verify that the DUT responds properly to the READ and NOP-Out PDUs by transmitting Data and Status=GOOD and NOP-In respectively.

Possible Problems: None.

Test #20.2: Data Digest Error on Immediate NOP-Out

Purpose: To see that the DUT properly handles a data digest error on an immediate NOP-Out PDU.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 13:13:04 2004

Discussion: At the target, any non-data PDU with a payload digest error is discarded and a Reject PDU is sent. The Initiator may then retry the PDU.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an immediate NOP-Out ping to the DUT. This NOP-Out should have a Data Digest Error.
· The DUT is expected to transmit a Reject PDU.
· The Testing Station should transmit a READ command to the DUT with a CmdSN the same as the rejected NOP-Out PDU, followed by an immediate NOP-Out PDU also with the same CmdSN as the original.

Observable Results:
· Verify that the DUT transmits a Reject PDU to the first immediate NOP-Out PDU. Verify that the DUT responds properly to the READ and immediate NOP-Out PDUs by transmitting Data and Status=GOOD and NOP-In respectively.

Possible Problems: None.

Test #21.1: Data Digest Error on Text Request

Purpose: To see that the DUT properly handles the recovery actions of an Initiator when a data digest error is detected on a Text Request PDU.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 13:44:23 2004

Discussion: At the target, any non-data PDU with a payload digest error is discarded and a Reject PDU is sent. The Initiator may then retry the PDU.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a Text Request to the DUT. This Text Request should have a Data Digest Error.
· The DUT is expected to transmit a Reject PDU.
· The Testing Station should transmit a READ command to the DUT with a CmdSN of one more than the rejected Text Request PDU, followed by a Text Request PDU with the same CmdSN as the original.

Observable Results:
· Verify that the DUT transmits a Reject PDU to the first Text Request PDU. Verify that the DUT responds properly to the READ and Text Requst PDUs by transmitting Data and Status=GOOD and Text Response respectively.

Possible Problems: None.

Test #21.2: Data Digest Error on Immediate Text Request

Purpose: To see that the DUT properly handles the recovery actions of an Initiator when a data digest error is detected on an immediate Text Request PDU.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 13:47:18 2004

Discussion: At the target, any non-data PDU with a payload digest error is discarded and a Reject PDU is sent. The Initiator may then retry the PDU.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an immediate Text Request to the DUT. This Text Request should have a Data Digest Error.
· The DUT is expected to transmit a Reject PDU.
· The Testing Station should transmit a READ command to the DUT with a CmdSN the same as the rejected Text Request PDU, followed by another immediate Text Request PDU with the same CmdSN as the original also.

Observable Results:
· Verify that the DUT transmits a Reject PDU to the first immediate Text Request PDU. Verify that the DUT responds properly to the READ and immediate Text Requst PDUs by transmitting Data and Status=GOOD and Text Response respectively.

Possible Problems: None.

Test #22.1: Data Digest Error on Text Response

Purpose: To see that the DUT properly handles the recovery actions of an initiator detecting a data digest error on a Text Response PDU.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 14:17:59 2004

Discussion: At the Initiator, any PDU with a payload digest error is discarded. The Initiator will then either request retransmission with SNACK or abort the task. The target then has the option of retransmitting the Data-In or transmitting SNACK Reject.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit a Text Request to the DUT, then wait for a Text Response.
· Once the Testing Station receives the Text Response, it should simulate a Data Digest Error on that Text Response. The Testing Station should transmit SNACK for the 'dropped' PDU.

Observable Results:
· Verify that the DUT either retransmits the PDU or transmits SNACK Reject.

Possible Problems: None.

Test #22.2: Data Digest Error on Text Response

Purpose: To see that the DUT properly handles the recovery actions of an initiator detecting a data digest error on a Text Response PDU when the Text request was immediate.

Reference: 6.1.4.2, 6.2, 6.7

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

Last Modification: Thu Feb 19 14:17:48 2004

Discussion: At the Initiator, any PDU with a payload digest error is discarded. The Initiator will then either request retransmission with SNACK or abort the task. The target then has the option of retransmitting the Data-In or transmitting SNACK Reject.

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

Procedure:
· Open a normal session to the DUT.
· The Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands, waiting for status on each.
· The Testing Station should transmit an immediate Text Request to the DUT, then wait for a Text Response.
· Once the Testing Station receives the Text Response, it should simulate a Data Digest Error on that Text Response. The Testing Station should transmit SNACK for the 'dropped' PDU.

Observable Results:
· Verify that the DUT either retransmits the PDU or transmits SNACK Reject.

Possible Problems: None.

Test #23.1: Connection Reinstatement

Purpose: To see that the DUT properly handlesan initiator reinstating a connection.

Reference: 5.3.4

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

Last Modification: Thu Feb 19 15:05:05 2004

Discussion: Connection reinstatement occurs when an initiator logs in with a target using a ISID-TSIH-CID combination that may currently active from the targets perspective.

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

Procedure:
· Open a normal session to the DUT.
· On two connections the Testing Station should perform a Standard Login, negotiating ImmediateData=No, InitialR2T=Yes. Negotiate ErrorRecoveryLevel>0.
· The testing station should perform INQUIRY, READ_CAPACITY, and TEST_UNIT_READY commands wait for status on each command.
· The Testing Station should drop one connection, then reinstate it using the same ISID, TSIH and CID.

Observable Results:
· Verify that the DUT accepts the reinstated connection and doesnt close the session.

Possible Problems: None.