TO: T10 Membership
FROM: Paul A. Suhler, Quantum Corporation
DATE: 3 November 2008
SUBJECT: T10/08-409r1, ADT-2: Internet ADT (iADT)

Revisions

T10/07-469:

0 Initial revision (2 November 2007)

1 First revision (9 March 2008)
Changed name to Network ADT (iADT).
Added registered port number.
Allowed iADT ports to use any port number.
Removed iADT-specific baud rate and $Timeout_{ACK}$.

2 Second revision (11 April 2008)
Deleted the ABORT service request and the ABORTED service indication.
Added analysis of existing state machines, link services, and frame header fields.
Added analysis of physical layer connections.

3 Third revision (30 April 2008)
Added discussion of including legacy ADT signals in new Custom Connector section.
Added proposed connector signal name and pinout.

4 Fourth revision (29 May 2008)
Separated the concept of an "ADT port" from an "ADT interconnect port."
Added link layer protocol services generic to all physical layers, as well as mappings to RS-422, TCP, and UDP.
Added requirement for ADT ports using Ethernet to ignore negotiated baud rate in computing acknowledgement timeout.

5 Fifth revision (13 June 2008)
Incorporated changes from 4 June 2008 teleconference, minutes in T10/08-256r0.
Enhanced model section.
Removed LED connections from ADT Ethernet bus description.
Added descriptions of LED blinking.
Specified a fixed $Timeout_{ACK}$ in seconds for ADT TCP connections.
Deleted ADT UDP interconnect.

6 Sixth revision (9 July 2008)
Incorporated changes from 18 June 2008 teleconference, minutes in T10/08-269r0.
Added definitions.
Modified layer figure.
Modified connection tables.
Added ladder diagrams for transport services.
Added background discussion of connection closing and I_T nexus loss.
Capitalize initial letters, per editor's guidance.

7 Seventh revision (25 July 2008)
Incorporated changes from 14 July 2008 meeting, minutes in T10/08-291r0.
Moved connection definitions into separate subclause from electrical characteristics.
Reorganized conventions subclauses to match SAM-4 and added a conventions subclause for ladder diagrams.
Revised ladder diagrams. (For connection establishment, used a single diagram with optional inter-device communication, rather than different diagrams for ADT serial and iADT ports as the working group had recommended.)

Changed terminology from ADT TCP interconnect port to iADT interconnect port.
Changed RS-422 references to match the actual number of the document (ANSI/EIA/TIA-422-B-1994).
Defined that deassertion of the $Sense_a$ or $Sense_d$ connection may invoke the **Closed** service indication and added a reason argument to **Closed**.
See T10/08-301r0 for a change to ADC-3 indicating that I_T nexus loss by bridging manager may be decoupled from command processing by local SMC device server.

8 Eighth revision (31 August 2008)
Incorporated changes from 13 August 2008 teleconference, minutes in T10/08-329r0.
This revision does not address any of the questions raised by IBM's T10/08-332r0:

- Support for short-lived connections, i.e., not having connection loss cause I_T nexus loss.
- Specifying what to do if a connection cannot be established.
- Relevance of the $Sense_a$ signal.
- Removing all references to Ethernet.
- Specifying signals to facilitate locating the drive in a library.

9 Ninth revision (6 October 2008)
Incorporated changes from 8 September 2008 meeting, minutes in T10/08-372r0.
Deleted retirement to send a Close event to Port state machine upon connection close.
Removed requirement to have $Sense_a$/$Sense_d$ asserted to establish a connection.
Incorporated comments from IBM's T10/08-392r0 and from an IBM e-mail about the Sockets API. This includes a new informative subclause 6.4.6.
Renamed Close service request and Closed service indication to Disconnect and Disconnected.
Added ADT Port argument to protocol services to identify the ADT port.
Deleted service responses that are generated only in response to invalid requests, e.g., coding errors like NULL arguments. Added subclause 6.2.11 and Table w indicating possible causes of other errors and recovery procedures.
Deleted some of the detailed renumbering instructions; the editor is capable of doing this himself.

T10/08-409:

0 Tenth revision (22 October 2008)
Incorporated changes from 8 October 2008 teleconference, minutes in T10/08-408r0.
Incorporated and expanded connection state machine from T10/08-407r1.
Included rules on invocation of service request in connection state descriptions.
Included use of sockets API function calls in connection state descriptions.
Adopted the term sADT port for ADT serial port.

1 Eleventh revision (3 November 2008)
Incorporated changes from an offline discussion with IBM.
Eliminated the concept of distinct ADT and ADT interconnect ports in favor of having an ADT port which may include multiple sessions.
Defined session to be an association between two ADT ports which is begun by a login and terminated by a logout (explicit or implicit). A session persists across multiple connections.
Modified layering diagrams to be consistent with SAM-4 Figure 31, Protocol service reference model.
For consistency with SAM-4, replaced connection layer "protocol services" with "connection services" and changed some indications into confirmations.
Added Sent connection service confirmation.
Split Disconnected service indication into Disconnected service confirmation and Disconnect Received service indication.

General

To allow future data transfer devices to have improved and alternate means to communicate with automation devices, Ethernet is proposed as an ADT port. One possible configuration would be an isolated subnet with the library controller and all drives attached. These ports will typically be 10/100BaseT, so there will be a great increase in bandwidth above the fastest existing RS422-based ADI ports.

Implementing an ADI Ethernet port could be done in two ways. One would be to use iSCSI to carry SCSI commands, data, and status and then to invent a new protocol for VHF data. A simpler approach would be to transport the entire ADT protocol over a networking protocol. This proposal is to do the latter, and is named Internet ADT (iADT).

A straightforward implementation of iADT would be to open a TCP connection between the automation device and the data transfer device. A TCP connection (also known as a stream) provides bi-directional reliable delivery of a stream of bytes. The existing ADT link layer protocol provides the necessary framing. While TCP error correction would prevent framing errors and parity errors from reaching the ADT layer, it would still be possible for acknowledgement timeouts to occur.

To avoid the need to modify ADT-2 to specify mapping of TCP connections to I_T nexuses, this proposal sidesteps the issue by stating that one ADT port connects to one other ADT port, without reference to the interconnect layer. At the interconnect layer, this proposal defines ADT interconnect ports through which ADT ports connect. There are two types of ADT interconnect ports, serial and Ethernet. One ADT serial port (sADT port) can connect only to one other sADT port, while multiple ADT Ethernet ports can connect to one another. Nevertheless, when ADT ports connect via ADT Ethernet ports, each ADT port can connect to only one other ADT port.

This organization of the standard avoids changes to the clauses for link, transport, and SCSI application layers.

Technical issues

The following are technical issues which must be considered in developing this proposal:

Timeouts

- After discussion in the May 2008 working group meeting, it was decided that the acknowledgement timeout should be used. While its use in detecting corrupted frames is not necessary when using TCP, it can still be used in recovering from a skip in frame numbers in at least one observed case. See the discussion below under ADT link layer analysis.

Negotiated Parameters

- Of the parameters in the Login IU, only Major/Minor Revision, Maximum Payload Size, and Maximum Ack Offset seem to be needed in iADT. Baud rate is unnecessary.

Port Numbers

- The original intent of this proposal was to use a fixed port number for the iADT port on both ends (sockets) of the TCP connection. A registered port number (4169) was obtained from the Internet Assigned Numbers Authority (IANA). However, existing Sockets implementations appear to dynamically assign the port number of the port performing a TCP active OPEN, so this requirement is relaxed. Instead, the only socket required to use 4169 is one in the device performing a passive OPEN (Listen). I.e, a DTD will do a passive OPEN on port 4169 and the library will connect to that port. Similarly, the library could do a passive OPEN on 4169 if it is desired for the DTDs to initiate the connection.
- If the network segment inside the library connects to a router that connects outside the library, then the drive can be protected by requiring the router not to pass packets with the

iADT port number in either the Source Port or Destination Port field of the TCP header. Requiring the receiving end of a connection request to use the iADT port number will facilitate this protection.

I_T Nexuses and TCP Connections

- With the advent of the session in 08-409r1, the I_T nexus is now defined as a session and the X_ORIGIN bit. For iADT, the session is defined to be a triple { local IP address, remote IP address, server port number }. These shall remain constant for the lifetime of the session.
- The need to include the server port number is disputed and must be resolved. The intent is to allow listening on different TCP port numbers. An alternative would be to mandate listening on port number 4169 for this standard; use of the iADT protocol to connect to other port numbers would be beyond the scope of this standard.
- There was a question whether the TCP ABORT could map to a device reset. David Black has since advised against this, saying "…an attempt to use this sort of TCP feature as a carrier of SCSI level function/semantics is not a good idea in general." Moreover, it is not clear (1) what events in a host already cause a TCP ABORT, and (2) whether the OS function to reset a storage device could be made to send an ABORT. Finally, RFC 793 specifies that an ABORT causes release of the TCB (control block), as does a CLOSE. This implies that an ABORT should also cause an I_T nexus loss.

Physical Layer

- The actual physical layer mandates Ethernet autonegotiation without mentioning specific speeds.

Custom Connector

The working group decided not to pursue a standard connector to include Ethernet. Instead, an ADT Ethernet "bus" is specified to list those connections which would be mandatory and optional.

- 1000BaseT requires four pairs of wires; usually all are wired in RJ-45 connectors and in Ethernet cables. However, 10 and 100BaseT only require two pair, so we discard the other two. There is no forecast need for an ADT Ethernet port to support Gigabit Ethernet.
- The ADT Ethernet bus will include the ADT $Sense_a$ line. Standalone DTDs may use Ethernet. Examples of how to discover presence in a library include a jumper or an extra pin on the Ethernet connector. If the DTD is not installed in a library, then it will enable its primary port(s) regardless of the saved setting of the port enable (PE) bit.
- Support for the $Reset_a$ connection is optional. In Ethernet, this will cause either a disconnection or a hard reset.
- In this revision of the proposal, support for the $Sense_d$ connection is optional.
- In this revision of the proposal, there is support for one or two LED connections to indicate Ethernet signal sense and activity. The connection will directly drive an LED which is pulled up via a resistor. The current and voltage characteristics of the connections are specified, but not those of the LED or resistor. This is intended to give designers maximum flexibility.
- The working group decided not to specify serial diagnostic connections in the ADT Ethernet bus

Discovery

- The working group wishes to specify how to discover the IP address of the library's and DTD's iADT ports.

- One possible means of discovery would be to use the Discovery and Description steps of the Universal Plug and Play (UPnP) protocol. This uses broadcast of UDP datagrams and does not require a server to track service locations. This would require the DTD to support an HTTP server. Proposal T10/08-198r0 describes how UPnP could be used. Discovery will not be a part of this proposal.

ADT link layer analysis

This section examines ADT's link-level specification for areas that are irrelevant to iADT, including frame header fields, information units, and state machines. While the current revision of the proposal makes no changes to the link layer, this information is retained for reference.

Much of the error recovery in ADT is to detect and correct physical-layer corruption of frames; these can be corrected by retransmitting the corrupted frame and are termed recoverable errors. Other errors, such as specifying an invalid protocol, setting a reserved bit, and sending a too-long packet can be due to firmware errors at a higher level. Simple retransmission cannot fix these errors and they are termed unrecoverable. TCP's reliable delivery will eliminate the recoverable errors, but cannot fix the unrecoverable errors.

State machines

The Transmitter Error and Receiver Error state machines are only used to recover from out of order or lost frames. TCP makes them unnecessary, and along with them the Initiate Recovery IUs.

Frame header fields

All of the frame header fields in ADT appear to be necessary in iADT. The following table summarizes the reasons.

Table 1 – Applicability of ADT frame header fields

Field	Comments
PROTOCOL	Needed to differentiate SCS Encapsulation, Fast Access, etc.
FRAME TYPE	Needed for various protocols
X_ORIGIN	Needed to distinguish exchanges originated by library from those originated by the DTD. This is effectively a part of the EXCHANGE ID field.
EXCHANGE ID	Needed to differentiate overlapped commands, etc.
FRAME NUMBER	Needed to associate ACKs and NAKs with frames.
PAYLOAD SIZE	Needed to help trap errors in frame assembly.

Timeouts

The original intent of the acknowledgement IU timeout in ADT was to recover from lost or corrupted (and thus discarded) frames. TCP should protect against both of these, so the only possible causes for this timeout would be slow processing in the receiver of the frame to be acknowledged or slow network transmission. However, a case was presented in which the acknowledgement timeout was used to recover from a malformed ACK IU. As a result, this revision of the proposal retains the acknowledgement timeout.

Link service IUs

Following is a summary of which ADT Link Service IUs are needed and which are not needed.

Table 2 – Applicability of ADT link service IUs

IU type	Comments
Login IU	Yes – Need a mechanism to agree on Major Revision, Minor Revision, Maximum Payload Size, and Maximum Ack Offset.
Logout IU	Yes – Need to provide logout duration and reason code.

Pause IU	Yes – This should probably be required before closing a connection.
NOP IU	No – Does anyone feel that this is needed?
Initiate Recovery IU	Yes
Initiate Recovery ACK IU	Yes
Initiate Recovery NAK IU	Yes
Device Reset IU	Yes
Timeout IU	Yes
ACK IU	Yes – While the flow control function of the ACK IU may not be needed, it still serves the purpose of indicating that a frame did not have non-recoverable errors. See the discussion below of the NAK IU.
NAK IU	Yes – See the following discussion of status codes.

The NAK IU is necessary to report certain errors that are due to an incorrectly-assembled frame; they are not related to corrupted or out-of-order frames. All of these errors are non-recoverable, i.e., they cannot be fixed by retransmission. For example, the upper layer assembling the frame may exceed the maximum payload length or may have a mismatch between the payload length field and the actual payload length.

Table 3 – Applicability of NAK IU status codes

Status code	Comments
OVER-LENGTH	Yes – This error can occur and cannot necessarily be corrected by retransmission.
UNDER-LENGTH	Yes – This error can occur and cannot necessarily be corrected by retransmission.
UNEXPECTED FRAME NUMBER	Yes – The ACK may be malformed.
AWAITING INITIATE RECOVERY IU	No
HEADER RESERVED BIT SET	Yes – This error can occur.
INVALID EXCHANGE ID	Yes – This error can occur.
UNSUPPORTED PROTOCOL	Yes – This error can occur.
OUT OF RESOURCES	Yes – This error can occur.
LOGIN IN PROGRESS	Yes – This error can occur.
INVALID OR ILLEGAL IU RECEIVED	Yes – This error can occur.
REJECTED, PORT IS LOGGED OUT	Yes – This error can occur
MAXIMUM ACK OFFSET EXCEEDED	Yes – This error can occur.
MAXIMUM PAYLOAD SIZE EXCEEDED	Yes – This error can occur.
UNSUPPORTED FRAME TYPE FOR SELECTED PROTOCOL	Yes – This error can occur.
NEGOTIATION ERROR	Yes – This error can occur
Vendor specific	Yes.

Editorial Notes

Paul Stone surveyed various T10 standards to determine how words in figures should be capitalized. The T10 style guide does not address this. Paul observed that the majority of standards capitalize initial letters, and requested that this proposal do likewise. Revision 6 incorporates this guidance; see also Revisions.

Revision 7 incorporates a conventions section for ladder diagrams. It also reorganizes the conventions section to more closely match that in SAM-4. However, it has retained the “state machine” terminology, while SAM-4 uses “state diagram.” See also Revisions.

Items Not Specified

The following technical issues have not been addressed in this proposal:

- While the maximum payload size decided on in ADT negotiation will continue to be driven by device resources, can it be kept independent of the TCP Maximum Segment Size (MSS), which is typically 1500 bytes in IPv4? An ADT frame split across multiple TCP segments might be handled inefficiently. (The MSS is the largest amount of data that can be sent in an unsegmented piece. The Maximum Transmission Unit (MTU) is the largest packet (header, data, and trailer) that can be sent. Because data is a component of a packet, MTU > MSS.)
- If a DTD is installed with both Ethernet and RS-422 ADI ports connected to the automation device, there could be confusion, although this would not be a new issue as currently nothing prohibits having two ADI ports. There is a practical issue, i.e., implementations may have taken shortcuts that would make the behavior of the ADC device server non-SAM-compliant with respect to multiple I_T nexuses. This is not a standards issue, and this proposal will not address the question of multiple ADI ports.
- Sockets APIs typically include an "out-of-band" channel that can be processed separately from regular data. This can be used to allow some data to bypass data sent earlier. This feature is not specified in this proposal, as it has no clear advantages and could potentially cause problems.

Changes to ADT-2 rev. 5

Markup conventions

Proposed additions are in blue, removed text is in ~~crossed out red~~.

Editor's notes in green provide information on decisions to be made and actions to be performed before this proposal can be integrated into the standard.

Change to clause 2

Add the following subclauses:

2.1.4 IETF references

RFC 791, *Internet Protocol – DARPA Internet Program – Protocol Specification*

RFC 793, *Transmission Control Protocol (TCP) – DARPA Internet Program – Protocol Specification*

RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*

RFC 3493, *Basic Socket Interface Extensions for IPv6*

2.1.5 IEEE references

IEEE 802.3-2005, *Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

2.2 Informative references

<…>

~~ANSI/TIA 422-B-1994 (R2000)~~ ANSI/EIA/TIA-422-B-1994 (revised January 27, 2000) Electrical Characteristics of Balanced Voltage Digital Interface Circuits. (RS-422)

Changes to clause 3

Add the following definitions:

3.1.x Connection: a means by which two ADT ports are able to exchange encoded characters (see 6.1).

3.1.x LLC: Ethernet link layer control.

3.1.x MAC: Ethernet media access control.

3.1.x MDI: Ethernet medium dependent interface.

3.1.x PHY: Ethernet physical layer.

3.1.x PLS: Ethernet physical signaling sublayer.

3.1.x Session: an association between two ADT ports existing after successful completion of link negotiation (see 4.3.3).

Reorganize clauses 3.4 through 3.6 as shown below and add a subclause for ladder diagram notation:

3.4 ~~Conventions~~ Editorial conventions

Certain words and terms used in this American National Standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 3 or in the text where they first appear. Names of signals, phases, messages, commands, statuses, sense keys, additional sense codes, and additional sense code qualifiers are in all uppercase (e.g., REQUEST SENSE), names of fields are in small uppercase (e.g., STATE OF SPARE), lower case is used for words having the normal English meaning.

Fields containing only one bit are usually referred to as the name bit instead of the name field.

~~Numbers that are not immediately followed by lower-case b or h are decimal values.~~

~~Numbers immediately followed by lower-case b (xxb) are binary values.~~

~~Numbers immediately followed by lower-case h (xxh) are hexadecimal values.~~

~~Decimals are indicated with a comma (e.g., two and one half is represented as 2,5).~~

~~Decimal numbers having a value exceeding 999 are represented with a space (e.g., 24 255).~~

An alphanumeric list (e.g., a,b,c or A,B,C) of items indicates the items in the list are unordered.

A numeric list (e.g., 1,2,3) of items indicate the items in the list are ordered (i.e., item 1 shall occur or complete before item 2).

In the event of conflicting information the precedence for requirements defined in this standard is:

1) text,
2) tables, then
3) figures.

3.5 Numeric conventions

Numbers that are not immediately followed by lower-case b or h are decimal values.

Numbers immediately followed by lower-case b (xxb) are binary values.

Numbers immediately followed by lower-case h (xxh) are hexadecimal values.

Decimals are indicated with a comma (e.g., two and one half is represented as 2,5).

Decimal numbers having a value exceeding 999 are represented with a space (e.g., 24 255).

3.6 Notation conventions

~~3.5~~ 3.6.1 Notation for ~~P~~procedures and ~~F~~functions

<…>

~~3.6~~ 3.6.2 ~~State machine conventions~~ Notation for state machines

~~3.6.1~~ 3.6.2.1 ~~State machine conventions overview~~ Notation for state machines overview

<…>

~~3.6.2~~ 3.6.2.2 ~~sub-state~~ Sub-state machines

<…>

~~3.6.3~~ 3.6.2.3 Transitions

<…>

~~3.6.4~~ 3.6.2.4 Messages, requests, and event notifications

<…>

3.6.3 Notation for communication sequence diagrams

Sequence diagrams are used to indicate communication among entities within a device and among devices. All communication sequence diagrams use the notation shown in Figure 3. Each entity is indicated by a horizontal bar with a label on top of a vertical bar. Entities within the same device are enclosed by a box with a label at the top of the box. Each communication is indicated by an arrow with an optional label. Solid arrows indicate mandatory communications and dashed arrows indicate optional communications. Time flows from the top of the diagram (i.e., first communication) to the bottom (i.e., last communication).

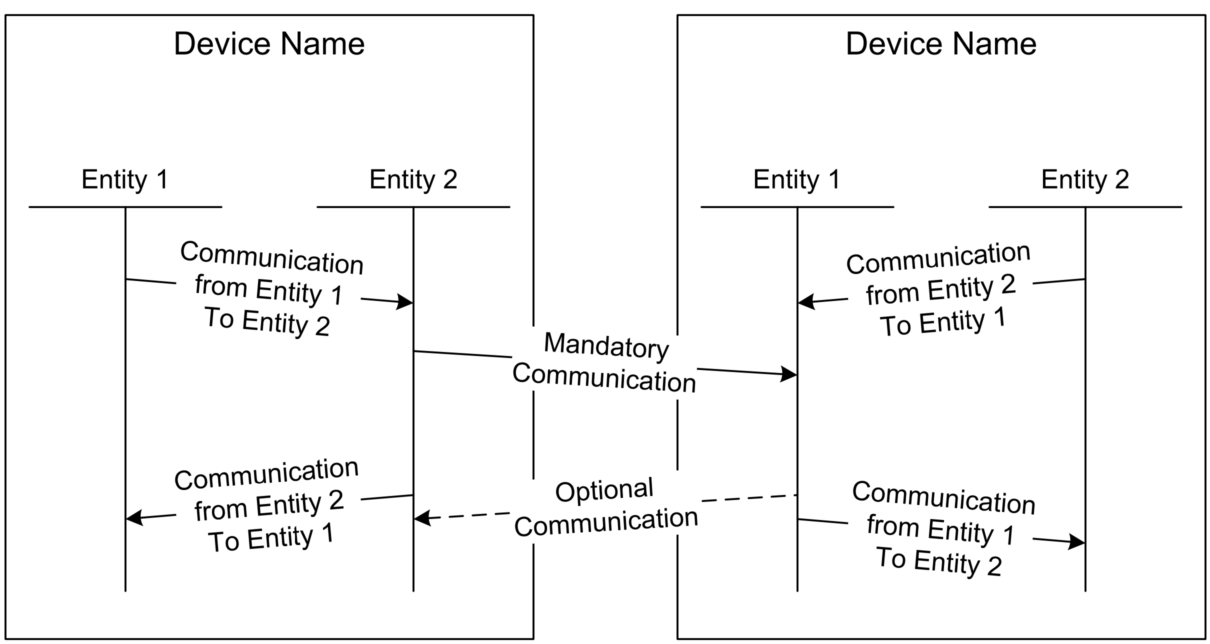

Figure 3 – Example communication sequence diagram

Changes to clause 4

Modify the beginning of clause 4.1:

4.1 Architecture

4.1.1 Architecture introduction

Figure ~~3~~ 4 shows an example of an ADT interface within a media changer containing two DT devices. Other common components of a media changer are also shown for reference. The components of an automation device are medium transport elements, data transfer (DT) devices, storage elements, and import/export elements (see SMC-3). The automation device communicates with the DT devices through ADT ports, as defined in this standard. DT devices and automation devices communicate with initiator ports other than those in the automation device using primary ports.

[Figure 3 is renumbered to 4]

If ADI Bridging is enabled (see ADC-2), each ADT port in the DT device and automation device is a SCSI target/initiator port. If ADI Bridging is disabled, the DT device port is a SCSI target port and the automation device port is a SCSI initiator port.

Add the following at the end of clause 4.1:

4.1.2 ADT protocol layers

The ADT protocol defines communication between two ADT ports. The ADT protocol includes the SCSI Transport Protocol Layer (STPL) and the Interconnect Layer (see SAM-4). The Interconnect Layer defined by ADT consists of three layers, the Session Layer, the Connection Layer, and the Physical Layer.

The interface between the SCSI application layer and the SCSI transport protocol layer is called the protocol service interface. The interface between the SCSI transport protocol layer and the interconnect layer is called the interconnect service interface. The interface between the session layer and the connection layer is called the connection service interface.

Figure 5 shows the communication between ADT ports at the different layers of the protocol, from the physical layer to the SCSI transport protocol layer.

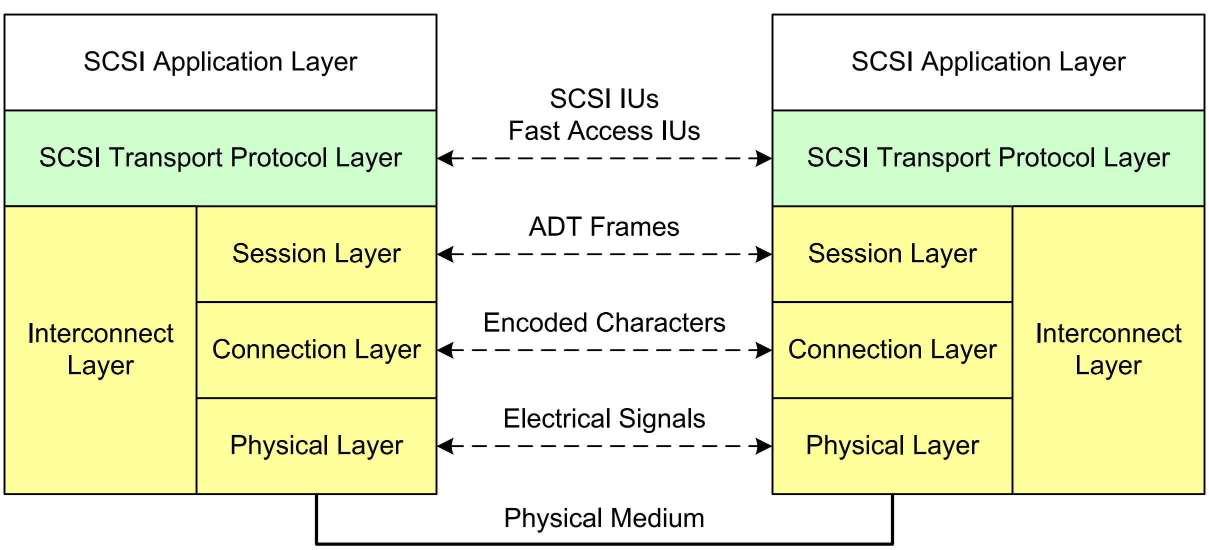

Figure 5 – ADT communication model

At the SCSI Transport Protocol Layer, information units (IUs) are passed between ADT ports. At the session layer, ADT frames are passed between ADT ports. At the connection layer, encoded characters are passed between ADT ports. At the physical layer, electrical signals are passed between ADT ports. The physical layers are connected by the physical medium.

Figure 6 shows the serial ADT (sADT) hierarchy of protocols which may be used to implement ADT on the RS-422 serial physical layer (see ANSI/EIA/TIA-422-B-1994 and 5.2.5.2).

ADT SCSI Encapsulation	ADT Fast Access	Transport Layer
ADT Link Layer		Session Layer
sADT		Connection Layer
RS-422		Physical Layer

Figure 6 – ADT serial protocol hierarchy

Figure 7 shows the Internet ADT (iADT) hierarchy of protocols which may be used to implement ADT on a physical layer supporting the Internet Protocol (IP), such as the Ethernet physical layer (see IEEE 802.3-2005 and 5.2.5.3).

ADT SCSI Encapsulation	ADT Fast Access	Transport Layer
ADT Link Layer		Session Layer
iADT		Connection Layer
TCP		
IP		
Ethernet LLC		
Ethernet MAC		
Ethernet PHY		Physical Layer

Figure 7 – iADT protocol hierarchy

The ADT physical layer (see clause 5) provides two alternative physical connections for data, RS-422 and Ethernet, as well as sense, signal, and LED connections.

The ADT connection layer (see clause 6) provides transmission of encoded characters between ADT ports. Two alternative transmission methods are defined, sADT and iADT. The sADT protocol provides transmission over an RS-422 physical layer. The iADT protocol provides transmission over a TCP connection (see RFC 793). The TCP connection uses the Internet Protocol (IP) (see RFC 791 and RFC 2460) to provide transmission over an Ethernet physical layer.

The ADT link layer (see clause 7) provides establishment of sessions between pairs of ADT ports and reliable transmission of ADT frames between the two ADT ports in a session. The ADT frames are represented as encoded characters.

The ADT transport layer (see clause 8) provides transmission of two categories of information units (IUs), SCSI encapsulation IUs and fast access IUs, between ADT ports. The information units are represented as ADT frames.

The SCSI application layer (see clause 9) provides transport protocol services for processing SCSI commands and task management requests.

The term sADT port refers to an ADT port using the ADT serial transmit-receive connections (see 5.2.5.2) and the sADT connection layer (see 6.3). An sADT port may connect to one other sADT port in another device. Figure 8 shows connections corresponding to Figure 4.

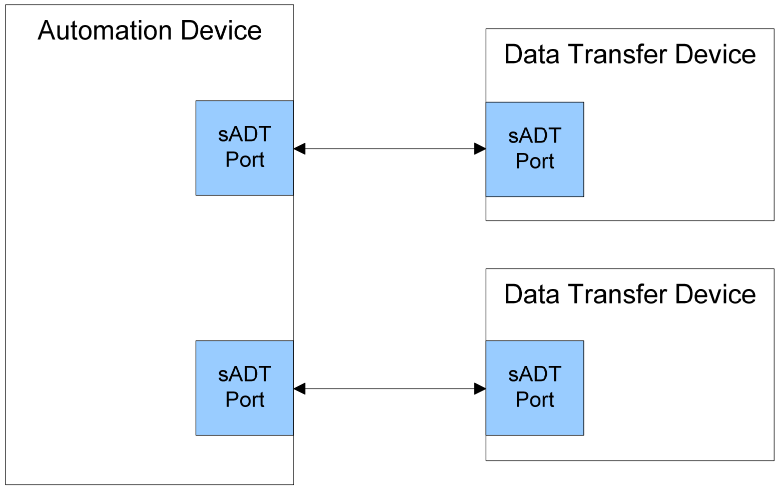

Figure 8 – sADT port example

The term iADT port refers to an ADT port using Internet Protocol (IP) transmit-receive connections, such as over Ethernet (see 5.2.5.3) and the iADT connection layer (see 6.4). An iADT port in one device may connect to multiple iADT ports in other devices. Figure 9 shows iADT ports connected via an IP network, corresponding to the connections shown in Figure 4.

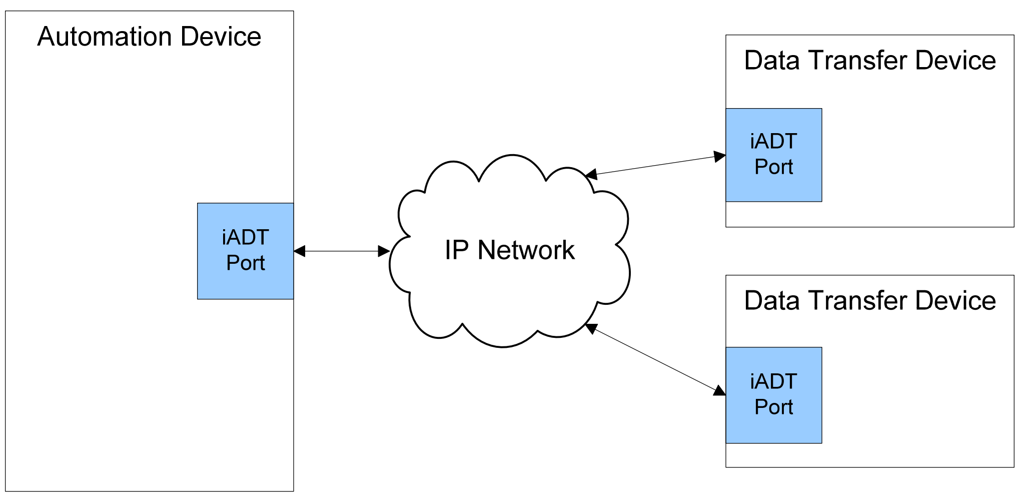

Figure 9 – iADT port example

4.3 ADT state machines

4.3.1 Introduction

The ADT transport layer contains ~~five~~ six state machines to manage ~~a connection~~ sessions and connections between two ADT ports. The state machines are as follows:

a) port;
b) link negotiation;
c) transmitter;
d) transmitter error recovery; ~~and~~
e) receiver error recovery~~.~~; and
f) connection.

The port state machine ~~is~~ and the connection state machine are the primary ~~machine~~ state machines and are always active. The other state machines are only active to manage specific operations (i.e they are sub-state machines of a state in the port state machine). An ADT port contains one copy of each state machine for each session. The connection state machine only exists in iADT ports

4.3.2 Port state machine
<…>
4.3.2.4 P2:Logged-In

4.3.2.4.1 State description

Upon entry to this state, a port shall set its operating parameters to the negotiated values (see 4.3.2.3.1).

While in this state, a session exists with the other ADT port with which link negotiation was performed

While in this state, the port's permission to transmit is managed through the use of the transmitter state machine.

<…>

4.3.7 Connection state machine

4.3.7.1 Connection state machine overview

The connection state machine is used in iADT ports to manage the connection process; it does not exist in sADT ports. The states are as follows:

a) C0:Not Connected;
b) C1:Listening;
c) C2:Connecting;
d) C3:Connected; and
e) C4:Disconnecting.

This state machine shall start in the C0:Not Connected state after a hard reset event.

Figure k shows the connection state machine. The following subclauses describe the transitions and the actions taken in each state.

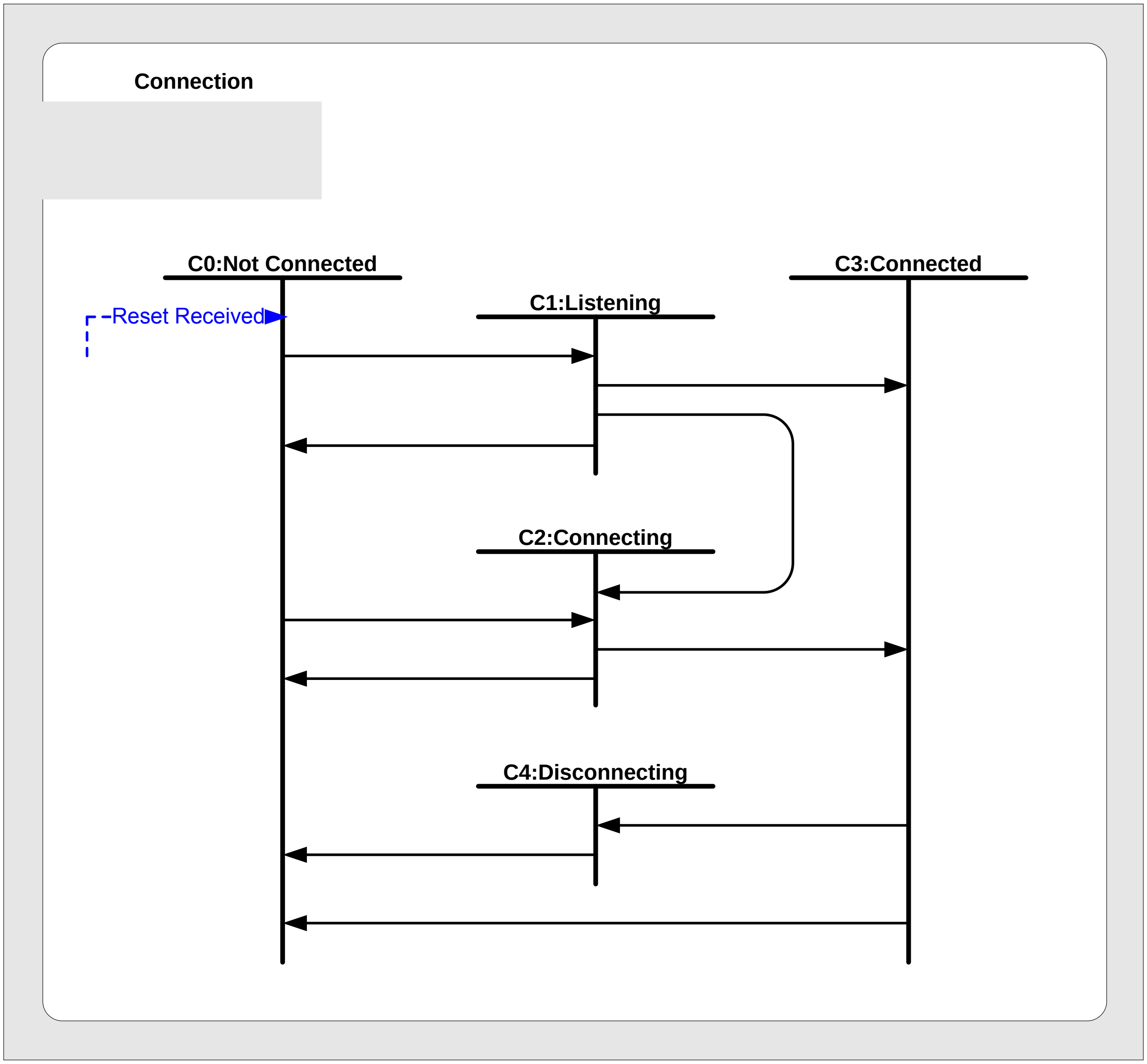

Figure k – Connection State Diagram

Note: We may want to add service requests as inputs, service indications as outputs, and other inputs to this figure.

4.3.7.2 C0:Not Connected state

4.3.7.2.1 State description

The C0:Not Connected state waits for the connection state machine to receive a **Listen** service request or a **Connect** service request (see 6.2).

When the iADT port enters the C0:Not Connected state from any other state, it shall release resources by invoking the close() function call.

While in the C0:Not Connected state, the iADT port shall:

a) accept **Connect** and **Listen** service requests from the ADT port;

a) reject **Send**, **Receive**, and **Disconnect** service requests (see 6.2) from the ADT port;
b) reject connection requests from remote iADT ports; and
c) discard encoded characters received from remote iADT ports.

4.3.7.2.2 Transition C0:Not Connected to C1:Listening

The iADT port shall transition to C1:Listening when it receives a **Listen** service request.

4.3.7.2.3 Transition C0:Not Connected to C2:Connecting

The iADT port shall transition to C2:Connecting state when it receives a **Connect** service request.

4.3.7.3 C1:Listening state

4.3.7.3.1 State description

The C1:Listening state waits for a connection request from a remote iADT port.

When the iADT port enters the C1:Listening state, the port shall perform a TCP passive OPEN (see RFC 793) by invoking the socket(), bind(), listen(), and accept() function calls.

While in the C1:Listening state, the iADT port shall:

a) accept **Connect** and **Disconnect** service requests (see 6.2)from the ADT port;
b) reject **Listen**, **Send**, or **Receive** service requests (see 6.2) from the ADT port;
c) accept connection requests from a remote iADT port; and
d) discard encoded characters received from a remote iADT port.

4.3.7.3.2 Transition C1:Listening to C3:Connected

The iADT port shall transition to the C3:Connected state and invoke a **Connected** service confirmation (see 6.2) when it accepts a connection from another ADT port (see RFC 793), as indicated by successful completion of the accept() function call.

4.3.7.3.3 Transition C1:Listening to C0:Not Connected

The iADT port shall transition to the C0:Not Connected state and invoke the **Disconnected** service confirmation (see 6.2) when it receives a **Disconnect** service request, as indicated by unsuccessful completion of the accept() function call.

4.3.7.3.4 Transition C1:Listening to C2:Connecting

The iADT port shall transition to the C2:Connecting state and release resources by invoking the close() function call, when it receives a **Connect** service request.

4.3.7.4 C2:Connecting state

4.3.7.4.1 State description

The C2:Connecting state attempts to connect to a remote iADT port.

When the iADT port enters the C2:Connecting state it shall perform a TCP active OPEN (see RFC 793) by invoking the socket(), bind(), and connect() function calls.

While in the C2:Connecting state, the iADT port shall:

a) accept **Disconnect** service requests from the ADT port;
b) reject **Listen**, **Send**, and **Receive** service requests from the ADT port;
c) establish a connection with a remote iADT port; and
d) discard encoded characters received from a remote iADT port.

4.3.7.4.2 Transition C2:Connecting to C0:Not Connected

The iADT port shall transition to the C0:Not Connected state and invoke the **Disconnected** service confirmation when:

a) it receives a **Disconnect** service request from the ADT port; or
b) the remote iADT port rejects the connection request, as indicated by unsuccessful completion of the connect() function call.

Note: Should this instead cause a Connected service confirmation with an error code, i.e., should Listen or Connect always result in Connected?

4.3.7.4.3 Transition C2:Connecting to C3:Connected

The iADT port shall transition to the C3:Connected state and invoke the **Connected** service confirmation when the remote iADT port accepts the connection request, as indicated by successful completion of the connect() function call.

4.3.7.5 C3:Connected state

4.3.7.5.1 State description

The C3:Connected state allows the ADT port to send and receive data.

When the iADT port enters the C3:Connected state, it shall invoke the **Connected** service confirmation.

While in the C3:Connected state, the iADT port shall:

a) accept **Send** service requests from the ADT port and transmit encoded characters to the other iADT port by invoking the send() function call;
b) accept **Receive** service requests from the ADT port and invoke the recv() function call;
c) receive encoded characters received from another iADT port and invoke the **Received** service confirmation upon successful completion of the recv() function call;
d) accept **Disconnect** service requests from the ADT port and invoke the shutdown() function call; and
e) reject **Connect** service requests from the ADT port.

4.3.7.5.2 Transition C3:Connected to C4:Disconnecting

The iADT port shall transition to the C4:Disconnecting state when it receives a **Disconnect** service request.

4.3.7.5.3 Transition C3:Connected to C0:Not Connected

The iADT port shall transition to the C0:Not Connected state when it receives a **Disconnect received** service indication.

4.3.7.6 C4:Disconnecting state

4.3.7.6.1 State description

The C4:Disconnecting state closes the connection with another iADT port. No further data may be accepted for transmission. Received data shall be accepted and passed to the ADT port.

When the iADT port enters the C4:Disconnecting state, it shall close the TCP connection by invoking the shutdown() function call.

While in the C4:Disconnecting state, the iADT port shall:

a) reject **Connect**, **Listen**, **Send**, and **Disconnect** service requests (see 6.2) from the ADT port;
b) accept **Receive** service requests (see 6.2) from the ADT port and invoke the recv() function call;
c) reject connection requests from other iADT ports; and
d) receive encoded characters received from another iADT port and invoke the **Received** service confirmation upon successful completion of the recv() function call.

4.3.7.6.2 Transition C4:Disconnecting to C0:Not Connected

The iADT port shall transition to the C0:Not Connected state and invoke the **Disconnected** service confirmation when the remote port closes the TCP connection, as indicated by unsuccessful completion of the recv() function call.

<…>

4.8 I_T nexus loss

An I_T nexus loss event shall occur if an ADT port:

a) sends a Port Login IU with the AOE bit set to one;
b) receives a Port Login IU with the AOE bit set to one;
c) receives an ACK IU in response to a Device Reset IU;
~~d) detects the change of state of the Sense line from presence to absence (i.e., $Sense_a$ for DT device port and $Sense_b$ for automation device port (see figure 11); or~~
~~e) detects the assertion of the $Reset_a$ line (see table 13).~~
d) receives a Reset service indication (see 6.6.10).

An I_T nexus loss may occur if an ADT port receives a Disconnected service indication from an iADT port.

Note: Permitting a nexus loss should be deleted, but must be discussed.

<…>

4.10.1 Acknowledgement time-out period calculation

When changing operating parameters (see 3.1.32), ~~a port~~ an ADT port connecting via an sADT port shall calculate a new acknowledgement IU time-out period using the formula in figure ~~9~~ 15. The port shall apply the new acknowledgement IU time-out period to every frame transmitted after changing operating parameters

Renumber Figure 9 to 15.

An ADT port connecting via an ADT Ethernet port shall use an initial acknowledgement time-out period of 2.5 seconds. This may be changed if the ADT port processes a time-out IU.

Changes to clause 5

5 Physical layer

5.1 Physical layer introduction

The ADT physical layer defines a number of connection types. Some of these connections are used by all ADT interconnect ports, some are used only by sADT ports, and some are used only by iADT ports. A connector is defined which may be used by sADT ports.

~~5.1~~ 5.2 Electrical characteristics

Modify Note 6 as follows:

NOTE 6 The connection specifications in sub clauses ~~5.1.3 through 5.1.5~~ 5.2.3, 5.2.4, and 5.2.5.2 assume cable with a R < 400 ohms/km, Z_0 = 100 ohms (nominal), and C = 50 pF/m (nominal).

Renumber Figure 10 to 16.

Modify clause 5.1.5 as follows:

~~5.1.5~~ 5.2.5 Transmit-receive ~~connection~~ connections

5.2.5.1 Transmit-receive connections introduction

This standard defines two sets of transmit-receive connections. The serial transmit-receive connection applies to implementations using the transmit-receive connections defined in 5.2.5.2. The Ethernet transmit-receive connection applies to implementations using Ethernet connections (see IEEE 802.3-2005).

5.2.5.2 Serial transmit-receive connections

A serial Transmit-Receive (Tx-Rx) connection is a complete simplex signal path from one ~~ADT~~ sADT port to a second ~~ADT~~ sADT port. A Tx-Rx connection includes:

a) a signal generator connected to the output compliance point of one ~~ADT~~ sADT port;
b) a pair of transmission media from the output compliance point of one ~~ADT~~ sADT port to the input compliance point of a second ~~ADT~~ sADT port; and
c) a signal receiver connected to the input compliance point of the second ~~ADT~~ sADT port.

A Tx-Rx connection shall conform to ~~TIA/EIA-422-B~~ ANSI/EIA/TIA-422-B-1994 as measured at the associated compliance points.

A Tx-Rx connection shall support 9 600 baud and may support the Modulation Rates listed in table 6.

A Tx-Rx connection shall use Non-return to Zero (NRZ) encoding of data bits to signaling elements. Hence, the data-signaling rate (in bps) equals the modulation rate (in baud).

A Tx-Rx connection shall transmit data bytes asynchronously adding one start bit, zero parity bits, and one stop bit to each data byte as depicted in figure ~~11~~ 17.

5.2.5.3 Ethernet transmit-receive connections

The electrical characteristics of Ethernet transmit-receive connections are defined in IEEE 802.3-2005.

Insert new clause 5.2.6:

5.2.6 LED connections

LED connections are used by a DT device to drive light-emitting diodes (LEDs) to indicate the status of the Ethernet connections. Table 7 describes the electrical characteristics of an LED connection at the output compliance point. The description assumes that:

a) the output is an open-collector type;
b) an LED and a resistor are connected in series between the output and the positive supply voltage.

Table 7 – LED connection output characteristics

Signal State	Current	Voltage
Asserted	-25 mA < I_{OL}	0 V < V_{OL} < 0.4 V
Negated	I_{OL} < 20 µA	V_{OH} < 5.5 V

Insert new clause 5.3 Connection names:

5.3 Connection instances

5.3.1 Sense connection instances

Table 8 defines the sense connections used by ADT interconnect ports:

Table 8 — Sense connections

Connection Name	O/M[a]	Connection Type	Driven By	Connection Definition
$Sense_a$	**O/M[b]**	Sense	automation device port	A DT device shall use this connection to sense the presence or absence of an automation device on the ADT bus.
$Sense_{aux}$	O	Sense		This standard does not define the use of this connection.
$Sense_d$	O	Sense	DT device port	An automation device shall use this connection to sense the presence or absence of a DT device on the ADT bus.
[a] O indicates support is optional; M indicates support is mandatory. [b] Mandatory for sADT ports. Optional for iADT ports.				

5.3.2 Signal connection instances

Table 9 defines the signal connections used by ADT interconnect ports:

Table 9 — Signal connections

Connection Name	O/M[a]	Connection Type	Driven By	Connection Definition
$Reset_a$	O	Signal	automation device port	An automation device may use this connection to signal a reset request to a DT device by invoking the Reset service request. A DT device shall treat the receipt of a signal on this connection as an invocation of the Reset Received service indication in the ADT port attached to the ADT bus (see 6.2.10).
$Signal_{aux}$	O	Signal		This standard does not define the use of this connection.
[a] O indicates support is optional; M indicates support is mandatory.				

5.3.3 Serial transmit-receive connection instances

Table 10 defines the transmit-receive connections for ADT serial interconnect ports.

Table 10 – ADT serial transmit-receive connections

Connection Name	O/M[a]	Connection Type	Driven By	Connection Definition
Tx_a - Rx_d	M	Tx-Rx	automation device port	An automation device shall use this connection to send serialized data. A DT device shall receive serialized data on this connection.
Tx_d - Rx_a	M	Tx-Rx	DT device port	A DT device shall use this connection to send serialized data. An automation device shall receive serialized data on this connection.
[a] O indicates support is optional, M indicates support is mandatory for ADT serial interconnect ports.				

5.3.4 Ethernet transmit-receive connection instances

Table 11 defines the transmit-receive connections for iADT interconnect ports.

Table 11 – Ethernet transmit-receive connections for Ethernet iADT ports

Connection Name	O/M[a]	Connection Type	Driven By	Connection Definition
TX_D1+	M	MDI [b]	[c]	See IEEE 802.3-2005.
TX_D1-	M	MDI [b]	[c]	See IEEE 802.3-2005.
RX_D2+	M	MDI [b]	[c]	See IEEE 802.3-2005.
RX_D2-	M	MDI [b]	[c]	See IEEE 802.3-2005.
BI_D3+	O	MDI [b]	[d]	See IEEE 802.3-2005.
BI_D3-	O	MDI [b]	[d]	See IEEE 802.3-2005.
BI_D4+	O	MDI [b]	[d]	See IEEE 802.3-2005.
BI_D4-	O	MDI [b]	[d]	See IEEE 802.3-2005.

[a] O indicates support is optional, M indicates support is mandatory for iADT interconnect ports.
[b] Medium Dependent Interface (MDI) and alternate MDI (MDI-X) are defined in IEEE 802.3-2005. An MDI connection shall support autonegotiation of link speed.
[c] In the MDI configuration, the port drives the TX_D1 pair. In the MDI-X configuration, the port drives the RX_D2 pair.
[d] The BI_D3 and BI_D4 pairs are driven as indicated by IEEE 802.3-2005.

5.3.5 LED connection instances

Table 12 defines the LED connections used by the DT device.

Table 12 – LED connections

Connection Name	O/M[a]	Connection Type	Driven By
LED_{active}	O	LED	DT device port
LED_{signal}	O	LED	DT device port
[a] O indicates support is optional, M indicates support is mandatory.			

A DT device supporting both the LED_{signal} and LED_{active} connections may signal in the following manner:

a) if carrier is detected (see IEEE 802.3-2005), the LED_{signal} connection is asserted. If no carrier is detected, the LED_{signal} connection is deasserted and the LED_{active} connection is deasserted; and

b) if data is being transmitted or received on the TX_D1 or RX_D2 connections (see IEEE 802.3-2005), the LED_{active} connection is alternately asserted and deasserted. If no data is being received on the TX_D1 or RX_D2 connections, the LED_{active} connection is deasserted.

A DT device supporting only the LED_{signal} connection may signal in the following manner:

a) if no carrier is detected, the LED_{signal} connection is deasserted;
b) if carrier is detected and no data is being received on the TX_D1 and RX_D2 connections, the LED_{signal} connection is asserted; and
c) if data is being received on the TX_D1 or RX_D2 connections, the LED_{signal} connection is alternately asserted and deasserted.

Renumber clause 5.2 Connector pin-out to 5.3.

~~5.3~~ 5.4 Connector pin-out

ADT serial interconnect ports ~~shall~~ may use the plug connector defined in SFF-8054. Table ~~8~~ 13 defines the pinout for the ADT port connector on the DT device.

Table ~~8~~ 13 – DT device ~~ADT~~ sADT port connector pinout

Pin Number	Connection Name	Reference
1	$+Tx_a$ - Rx_d	Table 16
2	$-Tx_a$ - Rx_d	Table 16
3	Ground	
4	$-Tx_d$ - Rx_a	Table 16
5	$+Tx_d$ - Rx_a	Table 16
6	$Sense_d$	Table 3
7	$Sense_a$	Table 3
8	$Reset_a$	Table 7
9	$Signal_{aux}$	Table 7
10	$Sense_{aux}$	Table 7

No connector pin-out is defined for the use of iADT ports.

New clause 6

Insert a new clause 6 between 5 (Physical layer) and 6 (Link layer):

6 Connection layer

6.1 Connection layer introduction

An ADT port shall establish a connection with another ADT port before transmitting or receiving encoded characters. After a connection is established, the connection may be closed and reopened later. Each connection is associated with one and only one session.

The ADT connection layer provides connection services for transmitting and receiving sequences of encoded characters between ADT ports. Table 14 summarizes the ADT connection services.

Table 14 – ADT connection services

Connection service	Connection service type	Invoked by
Listen	Request	Either
Connect	Request	Either
Connected	Confirmation	Either
Send	Request	Either
Sent	Confirmation	Either
Receive	Request	Either
Received	Confirmation	Either
Disconnect	Request	Either
Disconnected	Confirmation	Either
Disconnect received	Indication	Either
Reset	Request	Automation device
Reset received	Indication	DT device

The **Sense** service request determines whether the $Sense_a$ connection (in a DT device) or the $Sense_d$ connection (in an automation device) is asserted.

An ADT port may either initiate a connection to a specific ADT port, or await a connection from any ADT port. An ADT port initiates a connection by invoking the **Connect** service request. An ADT port awaits a connection by invoking the **Listen** service request. When the connection is established, both ADT ports receive a **Connected** service confirmation. The ADT ports may exchange information in order to establish the connection. To establish a connection sADT ports do not exchange information and iADT ports do exchange information.

Figure 18 shows an example of the relationships among the connection services used to establish a connection between two sADT ports. When closing a connection, no communication takes place between the devices.

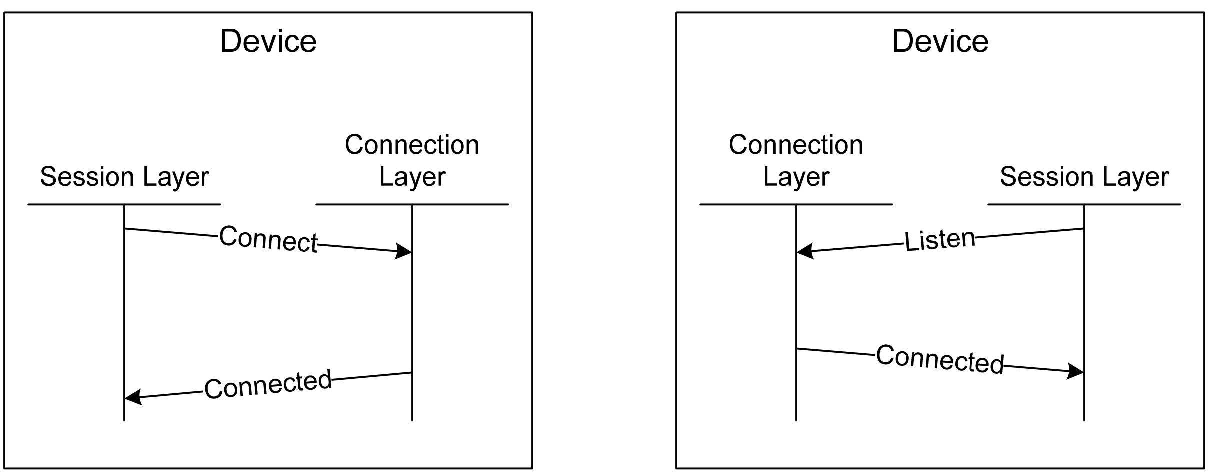

Figure 18 – Connection services for establishing a connection between sADT ports

NOTE n A connection is always considered to exist between a pair of sADT ports when the $Sense_a$ signal is asserted. For that reason, an sADT port invokes the **Connected** service confirmation immediately following the successful invocation of the **Connect** or **Listen** service request. The **Connected** service request may fail if the Sense connection is deasserted.

Figure 18a shows an example of the relationships among the connection services used to establish a connection between two iADT ports. The communication between the two devices is defined in RFC 793 and may constitute more than the two communications shown.

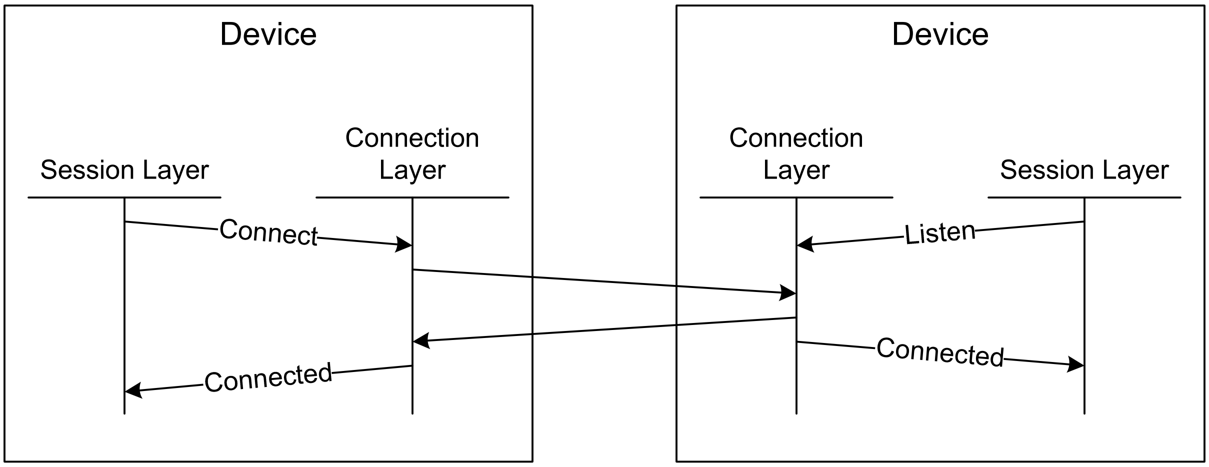

Figure 18a – Connection services for establishing a connection between iADT ports

Figure 19 shows the relationships among the connection services used to transfer data.

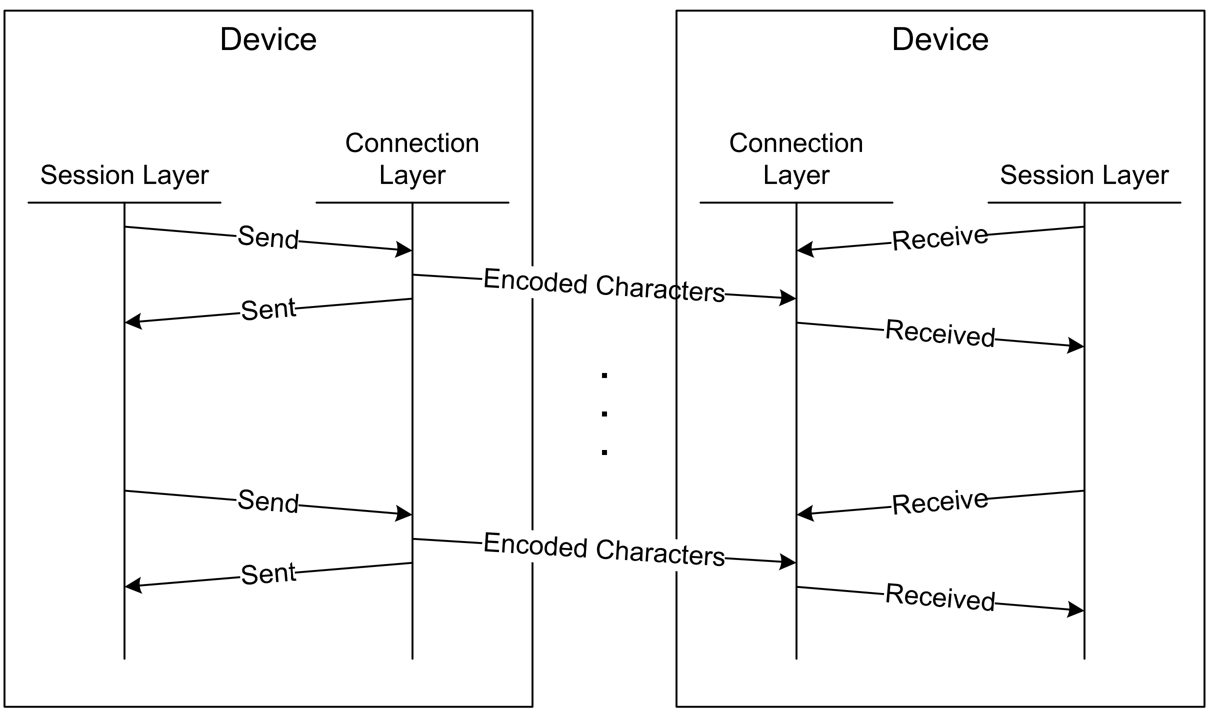

Figure 19 – Connection services for transferring data

An ADT port sends encoded characters on a connection by invoking the **Send** service request. The **Send** service request specifies the connection, the buffer containing the characters to be sent, and the number of characters to be sent. When the **Sent** service confirmation is invoked, the characters have been accepted by the Connect layer for delivery, and may have been transmitted, depending upon whether the port is an sADT or iADT port.

An ADT port receives encoded characters on a connection by invoking the **Receive** service request and then processing the **Received** service confirmation. The **Receive** service request specifies the connection, the buffer to contain the received characters, and the maximum number of characters to be placed in the buffer. When characters have been placed in the buffer, the **Received** service confirmation is invoked. The **Received** service confirmation indicates the number of characters that have been placed in the buffer. To receive more characters on the connection, the ADT port must invoke the **Receive** service request again.

Figure 20 shows the relationships among the connection services used to close a connection between two sADT ports. When closing a connection, no communication takes place between the devices.

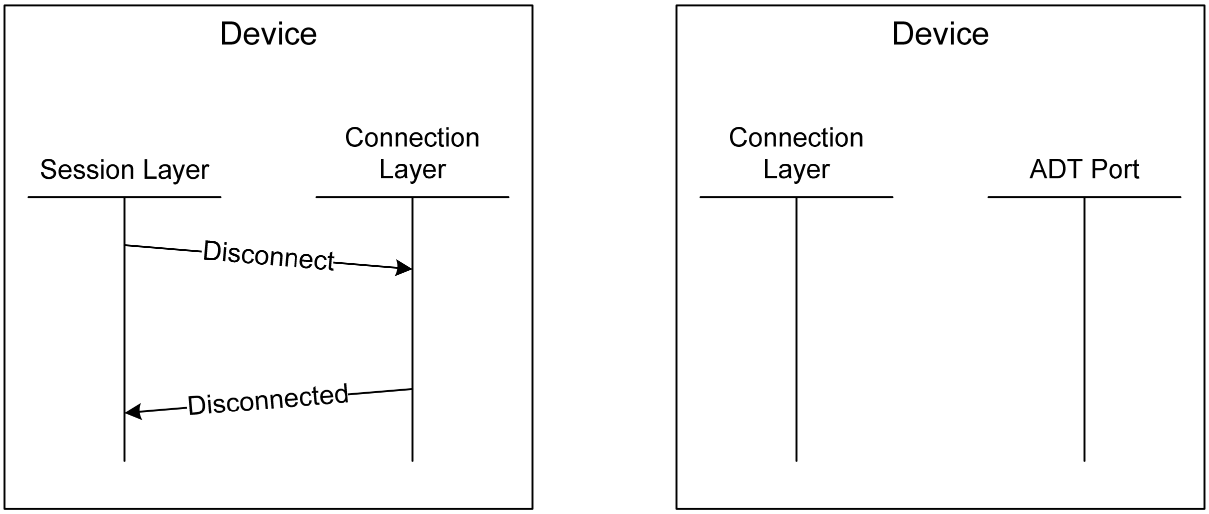

Figure 20 – Connection services for closing a connection between sADT ports

Figure 20a shows the relationships among the connection services used to close a connection between two iADT ports. The communication between the two devices is defined in RFC 793 and may constitute more than the two communications shown.

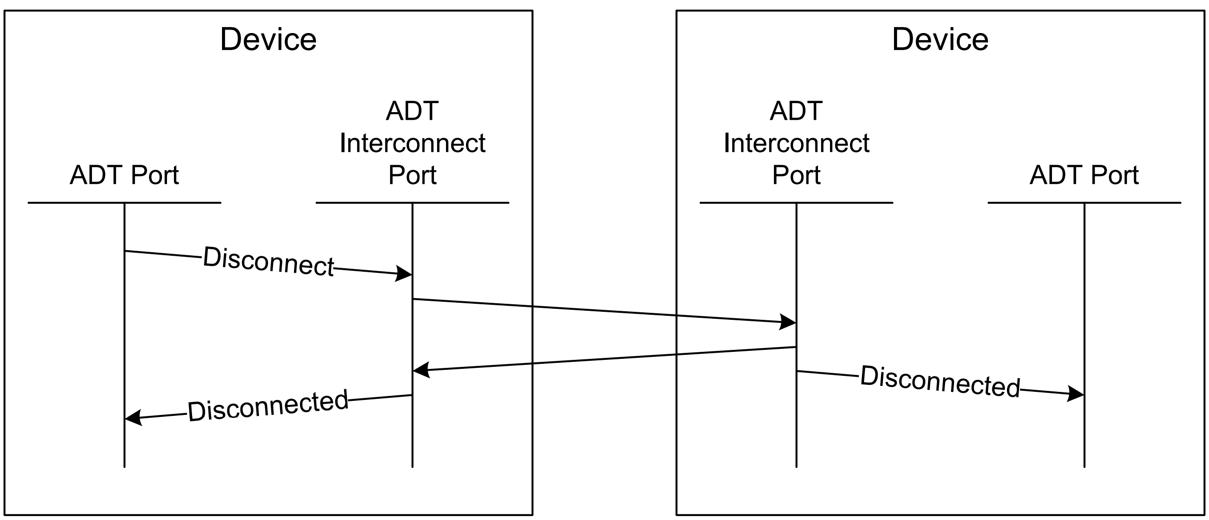

Figure 20a – Connection services for closing a connection between iADT ports

An ADT port closes a connection by invoking the **Disconnect** service request. Any characters that have been submitted for delivery by earlier **Send** service requests will be transmitted before the connection is closed. When an ADT port receives a **Disconnected** service indication, the connection is closed and no more characters shall be received. The ADT connection ports may or may not exchange information in order to close the connection. To close a connection ADT serial interconnect ports do not exchange information and iADT interconnect ports do exchange information.

Figure 21 shows the relationships among the connection services used to perform a reset.

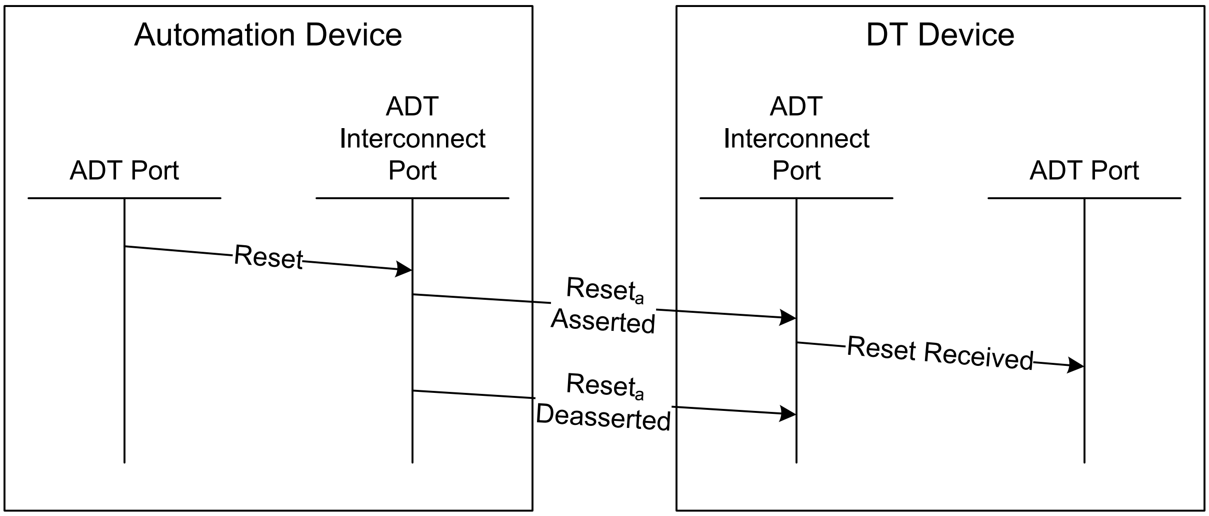

Figure 21 - Connection services for performing a reset

An ADT port in an automation device resets an ADT port in a DT device by invoking the **Reset** service request. The ADT port then asserts the Reset$_a$ connection. Assertion of the Reset$_a$ connection causes the ADT port in the DT device to receive a **Reset received** service indication.

6.2 Connection layer connection service definitions

Note: Some information in this subclause is redundant with that in the connection state machine and should be considered for removal.

6.2.1 Connect request

An ADT port uses the **Connect** connection service request to initiate a connection between a local ADT port and a specific remote ADT port.

An ADT port may use the **Connect** service request to exit C1:Listening state and enter C2:Connecting state.

An ADT port shall not invoke the **Connect** service request when it is in any state other than C0:Not Connected or C1:Listening.

An ADT port shall not invoke the **Connect** service request on a local interconnect port after it has invoked the **Disconnect** service request on that port and before it has received a **Disconnected** service indication.

Service Response = **Connect (IN (Session, Local Port, Remote Port))**

Input arguments:

Session: An identifier for the session with which the connection will be associated.

Local Port: An identifier for the local interconnect port.

Remote Port: The identifier for the remote interconnect port. This argument shall be ignored by an sADT port.

Service Response assumes one of the following values:

GOOD: The request completed successfully.

INVALID LOCAL PORT: The request failed because the **Local Port** argument did not specify a valid local interconnect port.

LOCAL PORT IN USE: The request failed because the **Local Port** argument specified a local interconnect port that was unable to support any more connections.

INVALID REMOTE PORT: The request failed because the **Remote Port** argument did not specify a valid remote interconnect port. See 6.2.11.

CONNECTION REFUSED: The request failed because the **Remote Port** did not accept the connection. This service response shall be reported if no ADT port had performed a **Listen** service request on the remote interconnect port. See 6.2.11.

NO PHYSICAL CONNECTION: The request failed because the Sense connection was not asserted or because the Ethernet iADT port did not detect a signal. See 6.2.11.

NOT IN ALLOWED STATE: The port was not in C0:Not Connected state and not in C1:Listening.

See Table w for error recovery procedures.

6.2.2 Listen service request

An ADT port uses the **Listen** connection service request to await a connection from a remote port.

If the **Remote Port** argument is present, then the ADT port shall accept a connection from only that remote port. If the **Remote Port** argument is not present, then the ADT port shall accept a connection from any remote port.

An ADT port shall not invoke the **Listen** service request when it is in any state other than C0:Not Connected.

If the $Sense_a$ connection in a DT device or the $Sense_d$ connection in an automation device is deasserted, then the **Listen** service request may return a service response of NO PHYSICAL CONNECTION.

Service Response = **Listen (IN (Session, Local Port, [Remote Port]))**

Input arguments:

Session: An identifier for the session with which the connection will be associated.

Local Port: An identifier for the local interconnect port

Remote Port: The identifier for the remote interconnect port. If this argument is not specified, then the ADT port will accept a connection from any remote port. This argument shall be ignored by an sADT port.

Service Response assumes one of the following values:

GOOD: The request completed successfully.

INVALID LOCAL PORT: The request failed because the **Local Port** argument did not specify a valid local interconnect port.

LOCAL PORT IN USE: The request failed because the **Local Port** argument specified a local interconnect port that was unable to support any more connections.

NO PHYSICAL CONNECTION: The request failed because the Sense connection was not asserted or because the Ethernet iADT port did not detect a signal. See 6.2.11.

NOT IN ALLOWED STATE: The port was not in C0:Not Connected state.

See Table w for error recovery procedures.

6.2.3 Connected service confirmation

An interconnect port uses the **Connected** service confirmation to notify the ADT port that the requested connection has been established.

An ADT interconnect port shall not invoke the **Connected** service confirmation if the ADT port has not invoked either a **Connect** or a **Listen** service request.

Connected (IN (Session, Connection, Remote Port))

Input arguments:

Session: An identifier for the session with which the connection is associated.

Connection: The identifier for the connection assigned by the connection layer.

Remote Port: The identifier for the remote interconnect port.

6.2.4 Send service request

An ADT port uses the **Send** service request to send data on a connection.

If an ADT port is in any state other than the C3:Connected state, it shall not invoke the **Send** service request.

If a subsequent **Send** service request is invoked before all of the data in the buffer specified by a previous **Send** service request, then the ADT interconnect port shall send all of the data in the buffer for the previous invocation before sending any data in the buffer of the subsequent invocation.

If the **Send** service request returns a service response of **OK**, then the ADT port may modify the contents of the buffer without affecting the data to be transmitted.

When the **Send** service request returns a service response of **OK**, then the characters may or may not have been transmitted on the physical connection.

If the **Send** service request is invoked after the **Disconnect** service request, then the **Send** service request shall be rejected with a service response of **INVALID CONNECTION**.

Service Response = Send (IN (Session, Connection, Buffer, Buffer Size))

Input arguments:

Session: An identifier for the session with which the connection is associated.

Connection: The identifier for the connection.

Buffer: A buffer containing data to be transmitted. The data in the buffer shall be encoded (see 7.2).

Buffer Size: The number of characters of encoded data to be transmitted on the connection.

Service Response assumes one of the following values:

GOOD: The request completed successfully.

INVALID CONNECTION: The request failed because the **Connection** argument did not specify an established connection. See 6.2.11.

INVALID BUFFER: The request failed because the **Buffer** argument did not specify a valid buffer.

OUT OF RESOURCES: The request failed because the interconnect port lacked resources to accept more characters for transmission. See 6.2.11.

NOT CONNECTED: The port was not in the C3:Connected state.

Note: Use NOT IN ALLOWED STATE instead of NOT CONNECTED?

See Table w for error recovery procedures.

6.2.5 Sent service confirmation

An ADT port invokes the **Sent** service confirmation to notify the ADT port that the characters specified by the **Send** service request have been accepted for transmission. In an sADT port, the **Sent** service confirmation also indicates that the characters have been transmitted by the physical port.

Sent (IN (Session))

Input arguments:

Session: An identifier for the session with which the connection is associated.

6.2.6 Receive service request

An ADT port invokes the **Receive** service request to receive data from a connection. The data received shall be processed as specified in clause 7.

If an ADT port is in any state other than the C3:Connected state, it shall not invoke the **Receive** service request.

If the **Receive** service request is invoked a second time before the **Received** service confirmation has been invoked, then the second **Receive** service request shall be rejected with a service response of **RECEIVE PENDING**.

Service Response = Receive (IN (Session, Connection, Buffer, Buffer Size))

Input arguments:

Session: An identifier for the session with which the connection is associated.

Connection: The identifier for the connection.

Buffer: A buffer to contain received data.

Buffer Size: The maximum number of characters of encoded data to be placed in the buffer.

Service Response assumes one of the following values:

GOOD:	The request completed successfully.
INVALID CONNECTION:	The request failed because the **Connection** argument did not specify an established connection. See 6.2.11.
INVALID BUFFER:	The request failed because the **Buffer** argument did not specify a valid buffer.
RECEIVE PENDING:	The request failed because the ADT port has invoked the **Receive** service request and the interconnect port has not yet invoked the **Received** service confirmation. See 6.2.11.
NOT CONNECTED:	The port was not in the C3:Connected state.

Note: Use NOT IN ALLOWED STATE instead of NOT CONNECTED?

See Table w for error recovery procedures.

6.2.7 Received service confirmation

An ADT interconnect port invokes the **Received** service confirmation to notify the ADT port that a number of characters have been received.

There is not a one-to-one correspondence between invocations of **Send** in one ADT port and invocations of **Received** in the other ADT port, i.e., the characters delivered in one invocation of **Received** may have been sent by one or more invocations of **Send**. Similarly, the characters sent in one invocation of Send may be delivered in one or more invocations of **Received**.

An ADT port shall not invoke the **Received** service confirmation after the ADT interconnect port has invoked the **Disconnected** service indication and before a new connection has been established.

The ADT interconnect port shall not invoke the **Disconnected** service indication until all received characters have bee transferred to the ADT port.

Received (IN (Session, Buffer, Received Character Count))

Input arguments:

Session:	An identifier for the session with which the connection is associated.
Buffer:	A buffer containing data received. The data in the buffer shall be encoded (see 7.2).
Received Character Count:	The number of characters received and placed in the buffer.

6.2.8 Disconnect service request

An ADT port invokes the **Disconnect** service request to close a connection.

If an ADT port is in any state other than the C1:Listening, C2:Connecting, or C3:Connected state, it shall not invoke the **Disconnect** service request.

Service Response = Disconnect (IN (Session, Connection))

Input arguments:

Session:	An identifier for the session with which the connection is associated.
Connection:	The identifier for the connection.

Service Response assumes one of the following values:

GOOD:	The request completed successfully.
INVALID CONNECTION:	The request failed because the **Connection** argument did not specify an established connection. See 6.2.11.
NOT IN ALLOWED STATE:	The port was not in the C1:Listening, C2:Connecting, or C3:Connected state.

See Table w for error recovery procedures.

6.2.9 Disconnected service confirmation

The **Disconnected** service confirmation notifies the ADT port that the connection has been closed. The ADT interconnect port shall not invoke the **Disconnected** service indication until all received characters have been transferred from the ADT interconnect port to the ADT port.

If an interconnect port in a DT device detects the transition of the $Sense_a$ connection from asserted to deasserted, it may invoke the **Disconnected** service indication. If an interconnect port in an automation device detects the transition of the $Sense_d$ connection from asserted to deasserted, it may invoke the **Disconnected** service indication. If an Ethernet iADT port detects loss of signal, it shall invoke the **Disconnected** service indication.

Disconnected (IN (Session, Reason))

Input arguments:

Session:	An identifier for the session with which the connection was associated.
Reason:	The reason that the connection was closed.

Reason assumes one of the following values:

DISCONNECT REQUESTED:	The sADT port processed a **Disconnect** service request.
CLOSED STATE:	The iADT port detected loss of the TCP connection (see RFC 793) but not loss of Ethernet signal.
SENSE DEASSERTED:	The interconnect port detected transition of the $Sense_a$ connection from asserted to deasserted.
LOSS OF SIGNAL:	The Ethernet iADT port detected loss of signal.

6.2.10 Disconnect received service indication

The **Disconnect received** service indication notifies the ADT port that the connection has been closed by the remote port. The ADT port shall not invoke the **Disconnect received** service indication until all received characters have been transferred to the session layer.

If an ADT port in a DT device detects the transition of the $Sense_a$ connection from asserted to deasserted, it may invoke the **Disconnect received** service indication. If an ADT port in an automation device detects the transition of the $Sense_d$ connection from asserted to deasserted, it may invoke the **Disconnect received** service indication. If an Ethernet iADT port detects loss of signal, it shall invoke the **Disconnected** service indication.

Disconnect received (IN (Session, Reason))

Input arguments:

Session: An identifier for the session with which the connection was associated.

Reason: The reason that the connection was closed.

Reason assumes one of the following values:

CLOSED STATE: The iADT port detected loss of the TCP connection (see RFC 793) but not loss of Ethernet signal.

SENSE DEASSERTED: The interconnect port detected transition of the $Sense_a$ connection from asserted to deasserted.

LOSS OF SIGNAL: The Ethernet iADT port detected loss of signal.

6.2.11 Reset service request

An ADT port in an automation device uses the **Reset** service request to reset the ADT interconnect port and assert the $Reset_a$ connection (see table 9).

Reset (IN (ADT Port, Local Port, Remote Port))

ADT Port: An identifier for the ADT port invoking the service request.

Local Port: An identifier for the local interconnect port.

Remote Port: The identifier for the remote interconnect port.

6.2.12 Reset received service indication

The **Reset received** service indication in a DT device indicates that the ADT interconnect port has been reset by assertion of the $Reset_a$ connection (see table 9).

Reset received (IN (ADT Port, Local Port))

ADT Port: An identifier for the ADT port receiving the service indication.

Local Port: An identifier for the local interconnect port.

6.2.13 Error recovery

Table w indicates possible causes for service responses other than **GOOD** and possible recovery procedures.

Table w – Service response error indication processing

Service response	Cause	Recovery procedure
INVALID REMOTE PORT	Remote port was disabled	Create new connection
NOT IN ALLOWED STATE	Invoked service request when in a state not appropriate for that request	Invoke **Disconnect** service request and retry service request
CONNECTION REFUSED	Remote port was not ready for connection	Retry connection
NO PHYSICAL CONNECTION	Physical interface problem	Not specified by this standard
OUT OF RESOURCES	Local port has not sent previous data	Retry send after a delay
INVALID CONNECTION	Connection was closed	Create new connection and retry operation
RECEIVE PENDING	The ADT port has invoked the **Receive** service request and the interconnect port has not yet invoked the **Received** service confirmation	Retry **Receive** service request after processing **Received** service confirmation

6.3 sADT port support of connection services

6.3.1 Connection establishment

When an sADT port invokes either the **Connect** or the **Listen** service request, the connection is considered to be established immediately. Invocation of either service request shall cause no transmission of data on the physical link. When either service request is invoked, the interconnect port shall invoke the **Connected** service confirmation. The **Connected** service confirmation may be invoked before the **Connect** or **Listen** service request has returned.

Table x shows how the arguments to the **Connect** service request are used by the sADT port.

Table x – Connect service request usage by sADT port

Argument	ADT serial implementation
Session	The identifier assigned by the session layer for the session with which the connection will be associated
Local Port	Used to select the physical port
Remote Port	Ignored

Table x+1 shows how the arguments to the **Listen** service request are used by the sADT port.

Table x+1 – Listen service request usage by sADT port

Argument	ADT serial implementation
Session	The identifier assigned by the session layer for the session with which the connection will be associated
Local Port	Used to select the physical port
Remote Port	Ignored

Table x+2 shows how the argument to the **Connected** service confirmation is set by the sADT port.

Table x+2 – Connected service confirmation usage by sADT port

Argument	ADT serial implementation
Session	The value of the Session argument to the **Connect** or **Listen** service request
Connection	The identifier assigned by the connection layer for the connection and used by subsequent service requests
Remote Port	Ignored

6.3.2 Data transmission

Table x+3 shows how the arguments to the **Send** service request are used by the sADT port.

Table x+3 – Send service request usage by sADT port

Argument	ADT serial implementation
Session	The identifier for the session with which the connection is associated
Connection	The value of the Connection argument returned by the **Connected** service indication
Buffer	The buffer containing data to be transmitted
Buffer Size	The number of characters in the buffer to be sent. The characters are encoded, i.e., the number includes Escape characters

When the **Send** service request is invoked, the sADT port shall invoke the **Sent** service confirmation after the encoded characters have been transmitted by the physical port. Table x+4 shows how the argument to the **Sent** service confirmation is used by the sADT port.

Table x+4 – Sent service confirmation usage by sADT port

Argument	sADT port implementation
Session	The value of the Session argument of the **Send** service request

6.3.3 Data reception

Table x+5 shows how the arguments to the **Receive** service request are used by the sADT port.

Table x+5 – Receive service request usage by sADT port

Argument	ADT serial implementation
Session	The identifier for the session with which the connection is associated
Connection	The value of the Connection argument returned by the **Connected** service indication
Buffer	The buffer to contain received data
Buffer Size	The maximum number of characters to be placed in the buffer

Table x+6 shows how the arguments to the **Received** service confirmation are set by the sADT port.

Table x+6 – Received service confirmation usage by sADT port

Argument	ADT serial implementation
Session	The identifier for the session with which the connection is associated
Buffer	The buffer containing the received data. The buffer shall be the same buffer specified in the previous invocation of the **Receive** service request.
Received Character Count	The number of characters placed in the buffer

6.3.4 Closing a connection

When an ADT port successfully invokes the **Disconnect** service request:

a) the interconnect port shall transmit all characters which had been delivered to the sADT port by previous invocations of the **Send** service request which completed successfully;
b) the sADT port may discard any characters received on the physical connection after the invocation of the **Disconnect** service request; and
c) if any characters have been received by the sADT port and not yet transferred to the ADT port, then the sADT port shall accept **Receive** service requests and invoke the **Received** service confirmation until all received characters have been transferred.

When all characters received on the sADT port have been transferred to the ADT port, then the sADT port shall invoke the **Disconnected** service indication. The **Disconnected** service indication may be invoked before the **Disconnect** service request completes.

Table x+7 shows how the argument to the **Disconnect** service request is set by the ADT port.

Table x+7 – Disconnect service request usage by sADT port

Argument	ADT serial implementation
ADT Port	The identifier for the session with which the connection is associated
Connection	The value of the Connection argument returned by the **Connected** service indication

Table x+8 shows how the argument to the **Disconnected** service indication is set by the sADT port.

Table x+8 – Disconnected service indication usage by sADT port

Argument	ADT serial implementation
Session	The identifier for the session with which the connection was associated
Reason	Either CLOSE REQUESTED or SENSE DEASSERTED.

6.3.5 Performing a reset

An automation device shall invoke the **Reset** service request to reset the ADT port in a DT device. Table x+9 shows how the argument to the **Reset** service request is used by the sADT port.

Table x+9 – Reset service request usage by sADT port

Argument	ADT serial implementation
ADT Port	Identifier for the ADT port invoking the service request
Local Port	Used to select the interconnect port in the automation device to transmit the reset to the DT device
Remote Port	Identifier for the remote interconnect port receiving the reset.

A DT device shall treat the invocation of the **Reset received** service indication either:

a) as a port logout (see 7.5.5); or
b) as a hard reset (see 4.7).

Table x+10 shows how the argument to the **Reset received** service indication is set by the sADT port.

Table x+10 – Reset received service indication usage by sADT port

Argument	ADT serial implementation
ADT Port	Identifier for the ADT port receiving the service indication
Local Port	Indicates the DT device interconnect port which received the reset from the automation device interconnect port

6.4 iADT port support of connection services

6.4.1 Connection establishment

If a session is not active when a connection is established (i.e., the iADT port is not in the P2:Logged-In state), the iADT port shall assign a local IP address, remote IP address, and server port number to be associated with the session.

When an iADT port invokes the **Connect** service request, it shall perform an active **OPEN** call (see RFC 793) with the foreign socket specified by the remote IP address and server port number associated with the session. The means by which the local iADT port learns the IP address and server port number of the remote iADT port is beyond the scope of this standard.

When an active **OPEN** call has successfully completed, each iADT port shall invoke the **Connected** service confirmation.

The iADT port may support more than one session.

An iADT port in a DTD shall invoke the **Listen** service request specifying a Local Port argument with a server port number of 4169 and shall support at least one session using that server port number.

Table y shows how the arguments to the **Connect** service request are used by the iADT port.

The Remote Port argument shall specify the IP address of the remote port and the server port number associated with the session.

Table y – Connect service request usage by iADT port

Argument	iADT port implementation
Session	The identifier for the session with which the connection will be associated
Local Port	The IP address of the **local port** argument to the **OPEN** call
Remote Port	The **foreign socket** argument to the **OPEN** call (IP address and port number)

When an ADT port invokes the **Listen** service request, an iADT port shall perform a TCP passive **OPEN** call (see RFC 793) with the local socket specified by the local IP address and server port number associated with the session. If the **Remote Port** argument is specified, then the foreign socket shall be specified using the value of the argument. If the **Remote Port** argument is not specified, then the foreign socket shall not be specified.

Table y+1 shows how the arguments to the **Listen** service request are used by the iADT port.

The Local Port argument shall specify the IP address of the local port and the server port number associated with the session.

Table y+1 – Listen service request usage by iADT port

Argument	iADT port implementation
Session	The identifier for the session with which the connection will be associated
Local Port	The **local port** argument to the **OPEN** call (IP address and port number)

Table y+2 shows how the arguments to the **Connected** service confirmation are set by the iADT port.

Table y+2 – Connected service confirmation usage by iADT port

Argument	iADT port implementation
Session	The value of the Session argument to the **Connect** or **Listen** service request
Connection	Connection identifier returned by the **Connect** service request
Remote Port	IP address of the remote port

6.4.2 Data transmission

When the **Send** service request is invoked, the iADT port shall invoke the **SEND** call (see RFC 793) with the **PUSH flag** argument set. Table y+3 shows how the arguments to the **Send** service request are used by the iADT port.

Table y+3 – Send service request usage by iADT port

Argument	iADT port implementation
Session	The identifier for the session with which the connection is associated
Connection	The value of the Connection argument returned by the **Connected** service indication
Buffer	**buffer address** argument to **SEND** call
Buffer Size	**byte count** argument to **SEND** call

When the **Send** service request is invoked, the iADT port shall invoke the **Sent** service confirmation. Invocation of the **Sent** service confirmation by the iADT port does not indicate that the characters have been transmitted by the physical port. Table y+4 shows how the argument to the **Sent** service confirmation is used by the iADT port.

Table y+4 – Sent service confirmation usage by iADT port

Argument	iADT port implementation
Session	The value of the Session argument of the **Send** service request

6.4.3 Data reception

Table y+5 shows how the arguments to the **Receive** service request are used by the iADT port.

Table y+5 – Receive service request usage by iADT port

Argument	iADT port implementation
Session	The identifier for the session with which the connection is associated
Connection	The value of the Connection argument returned by the **Connected** service indication
Buffer	**buffer address** argument to **RECEIVE** call
Buffer Size	**byte count** argument to **RECEIVE** call

Table y+6 shows how the arguments to the **Received** service confirmation are used by the iADT port.

Table y+6 – Received service confirmation usage by iADT port

Argument	iADT port implementation
Session	The identifier for the session with which the connection is associated
Buffer	**buffer address** argument to **RECEIVE** call
Received Character Count	The number of characters placed in the buffer

6.4.4 Closing a connection

When an ADT port successfully invokes the **Disconnect** service request, then the iADT port shall invoke the **CLOSE** call (see RFC 793). TCP guarantees that characters previously transferred with the **SEND** call shall be delivered before the connection is closed.

Table y+7 shows how the argument to the **Disconnect** service request is used by the iADT port.

Table y+7 – Disconnect service request usage by iADT port

Argument	iADT port implementation
Session	the identifier for the session with which the connection is associated
Connection	The value of the **Connection** argument returned by the **Connected** service confirmation

When an iADT port that had invoked the **Disconnect** service request enters the CLOSED state (see RFC 793), it shall invoke the **Disconnected** service confirmation. Table y+8 shows how the argument to the **Disconnected** service confirmation is set by the iADT port.

Table y+8 – Disconnected service confirmation usage by iADT port

Argument	iADT port implementation
Session	The identifier for the session with which the connection was associated
Reason	Either **DISCONNECT REQUESTED , CLOSED STATE,** or **SENSE DEASSERTED**

When an iADT port that had not invoked the **Disconnect** service request enters the CLOSED state (see RFC 793), it shall invoke the **Disconnect received** service indication. Table y+9 shows how the argument to the **Disconnect received** service indication is set by the iADT port.

Table y+9 – Disconnect received service indication usage by iADT port

Argument	iADT port implementation
Session	The identifier for the session with which the connection was associated
Reason	Either **DISCONNECT REQUESTED , CLOSED STATE,** or **SENSE DEASSERTED**

6.4.5 Performing a reset

An automation device shall invoke the **Reset** service request to reset the ADT port in a DT device. Table y+10 shows how the argument to the **Reset** service request is used by the iADT port.

Table y+10 – Reset service request usage by iADT port

Argument	iADT implementation
ADT Port	Identifier for the ADT port invoking the service request
Local Port	Used to select the interconnect port in the automation device to transmit the reset to the DT device
Remote Port	Identifier for the remote interconnect port receiving the reset.

A DT device shall treat the invocation of the **Reset received** service indication either:

a) as a **Disconnected** service indication (see 6.2.8) and may open a new connection; or
b) as a hard reset (see 4.7).

Table y+11 shows how the argument to the **Reset received** service indication is set by the iADT port.

Table y+11 – Reset received service indication usage by iADT port

Argument	iADT implementation
ADT Port	Identifier for the ADT port receiving the service indication
Local Port	Indicates the DT device interconnect port which received the reset from the automation device interconnect port

6.4.6 Relationship to Sockets API (informative)

In TCP/IP implementations, the TCP calls mentioned above are typically invoked via a Sockets application programming interface (API). The details of the Sockets API varies between implementations. This subclause describes the typical semantics of the Sockets API function calls and how the connection services may be mapped to those function calls.

Table y+12 describes the function calls in a typical Sockets API.

Table y+12 – Sockets API function calls

Function	Description
socket()	Creates a socket descriptor that represents a communication endpoint. The arguments to the socket() function tell the system which protocol to use, and what format address structure will be used in subsequent functions
bind()	Assigns a name to an unnamed socket that represents the address of the local communications endpoint, i.e., IP address and port number. When a socket is created with socket(), it exists in a name space (address family), but has no name assigned. bind() requests that name be assigned to the socket.
connect()	Assigns the name of the remote communications endpoint and a connection is established between the endpoints.
listen()	Enables the socket to accept a specified number of connection requests from remote sockets. Up to that number of requests may be queued on the socket; if additional requests are received before a queued request is removed, then the additional

	requests are rejected. When an accept() is invoked on a socket with queued requests, then one request is removed and an additional request may be queued.
accept()	Accepts a connection request on a socket that is listening for connections. A queued request is removed from the socket, a new socket is created for the connection, which is defined by a remote IP address and port number and the local IP address and port number. Further packets on that connection are routed to the new socket. The original socket may be used to accept additional connection requests. If no connection request is queued on the original socket, then the accept() may block until one arrives or until a close() is invoked on the socket.
send()	Sends outgoing data on a connected socket.
recv()	Receives incoming data that has been received by a connected socket.
shutdown()	Closes a connection, optionally preventing further sends and/or receives.
close()	Deletes a socket descriptor created by the socket() function. If the socket was connected, the connection is terminated. Data that has yet to be delivered to the remote endpoint is discarded. To ensure transmission and reception of all pending packets, close() should be invoked after shutdown() has returned. If the deleted socket was the original one upon which the listen() was invoked, then no new connections can be accepted. Existing connections are unaffected.

Table y+13 shows how connection services may be mapped to Sockets API function calls. The **Reset** service request and **Reset received** service indication are not listed because they are not relevant to the Sockets API.

Table y+13 – Connection service mapping to Sockets API functions

Connection service	Socket function	Notes
Connect	socket()	
	bind()	bind() specifies a dynamic local port number.
	connect()	connect() specifies the remote socket address. This socket may not be reused for additional connections. Creating another connection requires invoking socket() to allocate a new socket resource and then invoking bind() and connect() on that new socket.
Listen	[socket()]	socket() is invoked if no prior **Listen** service request has invoked socket() for this local port.
	[bind()]	bind() is invoked if no prior **Listen** service request has invoked bind() for this local port. bind() may specify the iADT port number (4169).
	[listen()]	listen() is invoked if no prior **Listen** service request has invoked listen() for this local port. listen() enables the socket to accept one or more simultaneous connections.
	accept()	accept() may be invoked multiple times after a single listen(). Each invocation of **Listen** invokes accept() exactly one time.
Connected		accept() may block until a remote socket connects to the local socket. If so, then it returns the address of the remote socket. This return causes invocation of the **Connected** service confirmation, which returns the address of the remote socket in the Connection argument.

Send	send()	Invocation of the **Send** service request causes invocation of send().
Sent		The **Sent** service confirmation shall be invoked after the send() returns.
Receive	recv()	Invocation of the **Receive** service request causes the invocation of recv(), which will block until a message is received.
Received		recv() returns when a message is received. This return causes the invocation of the **Received** service confirmation.
Disconnect	shutdown()	
	close()	
Disconnected		After close() returns, the **Disconnected** service confirmation is invoked. Other events, e.g., physical port failure, may also cause invocation of **Disconnected**.
Disconnect received		The **Disconnect received** service indication shall be invoked after the iADT port is notified that the remote port has closed the connection.

Changes to clause 6

Renumbered to clause 7

Note 1: Does the session concept need to be explained here?
Note 2: Should we require issuing the Pause IU before performing a Close if there has been no Logout?
Note 3: Should we recommend against closing the connection to a drive performing bridging?

Changes to clause 7

~~7~~8 Transport layer

~~7~~8.1 SCSI Encapsulation

~~7~~8.1.1 SCSI encapsulation overview

SCSI information units contain information required to implement the SCSI protocol. Information units are routed to the remote ADT port associated with the session. The X_ORIGIN bit in the ADT frame header conveys the SCSI initiator port and SCSI target port identities. The EXCHANGE ID value from the ADT frame header of an encapsulated SCSI protocol IU takes on the role of the task tag from SAM-3. The LUN is included in the SCSI Command IU and SCSI Task Management IU payload contents. See 4.9 for transport protocol variations from SAM-3. See clause 8 for the mapping of the IUs described in this clause to the SCSI transport protocol services.

<…>

~~7~~8.2 Fast Access

~~7~~8.2.1 Fast Access overview

This protocol is intended to provide a feature set beyond what is provided by SAM-3 to both take advantage of the features of the transport layer and work around its slower speed. The Fast Access protocol provides:

a) a simple method for accessing the Very High Frequency (VHF) Data defined in ADC-2;
b) an asynchronous event report, a method for a DT device to report asynchronous activity; and
c) a method to control these asynchronous reports.

Fast Access protocol IUs are routed to the remote ADT port associated with the session.

<…>

Changes to clause 8

~~8~~9 SCSI application layer

<…>

~~8~~9.2 Transport layer protocol services to support Execute Command

<…>

Table 32 — Send SCSI Command transport layer protocol service arguments

Argument	ADT Implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I_T is used to select the session and to set the X_ORIGIN field; b) L is used to set the LUN field; and c) Q is set by the transport layer.
<…>	

<…>

Table 33 — SCSI Command Received transport layer protocol service arguments

Argument	ADT Implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I_T is indicated by the session and the X_ORIGIN field; b) L is used to set the LUN field; and c) Q is set by the transport layer.
<…>	

<…>

Table 35 — Command Complete Received transport layer protocol service arguments

Argument	ADT Implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I_T is indicated by the session and the X_ORIGIN field; b) L is used to set the LUN field; and c) Q is set by the transport layer.
<…>	

<…>

Table 40 — Send Data-Out transport layer protocol service arguments

Argument	ADT Implementation

I_T_L_Q nexus	Used to select the session and to set the X_ORIGIN and EXCHANGE ID fields in the ADT frame(s) header.
<…>	

<…>

Table 42 — Send Data-Out transport layer protocol service arguments

Argument	ADT Implementation
I_T_L_Q nexus	Used to select the session and to set the X_ORIGIN and EXCHANGE ID fields in the ADT frame(s) header.
<…>	

<…>