TO: T10 Membership
FROM: Paul A. Suhler, Quantum Corporation
DATE: 11 April 2008
SUBJECT: T10/07-469r2, ADT-2: Internet ADT (iADT)

Revisions

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.

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).

In this revision, the name of the protocol is changed from ADToE to iADT for consistency with other protocols. iADT is closer to iSCSI in that both run over TCP connections, and unlike Fibre Channel over (FCoE), which is mapped to the Ethernet protocol.

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.

Technical issues

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

Timeouts

- The acknowledgement timeout should not be used. 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. I.e, a DTD will do a passive OPEN on port 4169 and the library will connect to that. 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

- Existing ADT inherently has a single I_T nexus. For iADT, an I_T nexus will correspond to a TCP connection, of which there could be more than one. Each end of a TCP connection is a socket, which is specified by a unique combination of an IP address and a port number. One socket can connect to multiple other sockets. See the proposed changes below for more detail.
- By using multiple local sockets, i.e., port numbers, the library can open multiple connections to a drive's iADT port. Each of those connections would represent a different I_T nexus. Support in DTDs for multi-initiator ADT operation is beyond the scope of this proposal.
- There is a new case in the I_T nexus loss case list, for closing of the TCP connection while still logged in at the ADT level.
- A CLOSE on a connection will terminate transmissions from the sender of the CLOSE, but not in the other direction. The other side can continue to send. It may be necessary to address this behavior in this proposal.
- 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

- References to physical layer signals outside the physical layer clause have been qualified so that it will be obvious that they do not apply to iADT ports.
- The working group wishes to specify a standard small Ethernet connector for use on the DTD. A request has been made to connector vendors for input.
- The actual physical layer mandates Ethernet autonegotiation without mentioning specific speeds.
- The working group will consider whether there should be a means for the DTD to discover whether it is in a library, as is provided by 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.

- There has been a request to support the $Reset_a$ connection. In ADT, this connection could cause either a port logout or a hard reset. This requires further discussion.
- The other function provided by the ADT physical layer is the library's detecting the DTD, via the $Sense_d$ connection. This is probably not required. Automation vendors should comment.

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 packets and does not require a server to track service locations. This would require the DTD to support an HTTP server.

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.

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. To avoid this – and because the basic reason for the timeout does not apply – the acknowledgement timeout should not be used.

Link service IUs

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

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	No – If no receive() is performed on the connection, then data will not be lost. (This was originally intended to prevent dropping bytes on an RS-422 connection that was being ignored.)
NOP IU	No – Does anyone feel that this is needed?
Initiate Recovery IU	No – TE/RE state machines are not required.
Initiate Recovery ACK IU	No – TE/RE state machines are not required.
Initiate Recovery NAK IU	No – TE/RE state machines are not required.
Device Reset IU	Yes
Timeout IU	No – Timeouts are not required. (See **Timeouts** above.)
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	No – Assuming that we can ignore frame numbers because frames are delivered in order.
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 – Unless we can abolish logins.
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 – Unless we can abolish logins.
Vendor specific	Yes.

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.
- Should UDP be used in place of TCP? ADT error detection and recovery would then have to cope with loss and out-of-order delivery of packets. Would this improve performance (in the absence of errors)? Upon the initial discussion, most of the working group preferred not to use UDP.
- 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 case problems.

Documentation issues

The following are issues about how to integrate this new material into ADT-2’s structure:

- This proposal does not add a major subclause to the model section. Would it work better if there were one?
- The term “link layer” as used in ADT clashes with the Ethernet meaning of that term, since the ADT link layer will run on top of TCP. If TCP were added to ADT-2 as an alternate physical layer then the terminology would become even more confusing. The approach in this proposal is to add it as a top-level clause after the physical layer, and to add text where required to indicate that either the existing ADT physical layer in clause 5 or the TCP layer in the new clause 6 will apply to any given ADT port. An alternative would be to abstract the interface to the layer below the ADT link layer, which would make the ADT physical layer and the TCP-based ADT-over-Ethernet layer peers in the protocol hierarchy.
- RFC 793 provides an API for function calls from the user layer into the TCP layer. However, it does not specify function calls from TCP up into the user layer. Therefore this proposal specifies a complete API. iADT service requests are mapped to TCP calls, with the arguments to the iADT service requests being a subset of those of the TCP calls. Calls up into the user layer are specified as iADT service confirmations and iADT service indications.

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*

2.1.5 IEEE references

IEEE 802.3i-1990, *Supplement to 802.3 – 10Base-T*

IEEE 802.3aa-1998, *100BaseT Maintenance Revision #5*

Changes to clause 3

Add the following definition:

3.1.x ADT physical layer: The physical layer described in clause 5.

3.1.y iADT port: An ADT port that uses the transport layer described in clause 6.

3.1.z iADT TCP layer: The transport layer described in clause 6.

Changes to clause 4

Modify the first lettered list as follows:

Figure 4 – Port State Machine Diagram

Add an arrow labeled CLOSE Event (to all states, causing transition to P0:Initial) to upper left of figure.

4.6.1.2.2 Acknowledgement IU time-out

In a port that uses the ADT physical layer, the ~~The~~ sender of a frame, other than an acknowledgement IU, shall time-out the resulting acknowledgement. It shall be considered an error condition if a corresponding acknowledgement IU is not received within the time-out period. The time-out period shall start after the EOF of the frame has been sent. When operating with a maximum ACK offset greater than one, a port may start the time-out period for a frame that has completed transmission after the acknowledgement IU for a previously sent frame has been received.

In an iADT port, the sender of a frame shall not time-out the resulting acknowledgement. In an iADT port, the contents of the CURRENT ACKNOWLEDGEMENT TIME-OUT, MAXIMUM ACKNOWLEDGEMENT TIME-OUT, MINIMUM ACKNOWLEDGEMENT TIME-OUT, and TIME-OUT RESOLUTION fields in the Time-out IU (see Table 18) shall be ignored.

4.7 Hard reset

A hard reset is a response to a power on condition or optionally, for an ADT port that uses the ADT physical layer, a $Reset_a$ event (see table 7). The target port's response to a hard reset shall include initiating the equivalent of a logical unit reset for all logical units as described in SAM-3.

The effect of the hard reset on tasks that have not completed, SCSI device reservations, and SCSI device operating modes is defined in SAM-3.

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) that uses the ADT physical layer 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~~ 12); ~~or~~
e) that uses the ADT physical layer detects the assertion of the $Reset_a$ line (see table ~~7~~ 13)~~.~~; or
f) that uses the TCP layer receives a CLOSED service indication while the Port State Machine is in either P1:Login or P2:Logged In state (see 6.3.2).

4.10.1 Acknowledgement time-out period calculation

When changing operating parameters (see 3.1.32), a port that uses the ADT physical layer shall calculate a new acknowledgement IU time-out period using the formula in figure 9. The port shall apply the new acknowledgement IU time-out period to every frame transmitted after changing operating parameters.

Change to clause 5

Add a new subclause 5.1:

5 Physical layer

5.1 Physical layer applicability

This clause does not apply to iADT ports.

Subsequent subclauses are re-numbered to 5.2, 5.3, and 5.4.

New clause 6 for TCP connections

Add the following clause after the existing clause 5, Physical layer:

6 iADT TCP layer

6.1 iADT TCP layer introduction

This clause does not apply to ADT ports that use the ADT physical layer.

Encoded ADT frames may be transmitted and received between two ADT ports via a Transmission Control Protocol (TCP) network connection (see RFC 793), rather than by the ADT physical layer. A TCP connection provides a bi-directional stream of bytes that are guaranteed to be delivered in the order in which they were sent. The iADT TCP layer described in this clause defines how ADT ports use TCP connections to send ADT frames. The ADT link layer protocol (see 7) provides framing, error detection, and error correction on the byte streams sent and received via the TCP layer.

A TCP connection is uniquely identified by a pair of sockets (see RFC 793). A socket is defined as an internet address concatenated with a port identifier. The internet address is either an IPv4 (see RFC 791) or IPv6 (see RFC 2460) address. An iADT connection is a TCP connection used to transmit ADT frames. An iADT port is an ADT port that uses an iADT connection to communicate with another ADT port.

Within an automation or DT device, multiple iADT ports may connect to remote iADT ports via the same local iADT socket. Within a device, an ADT port shall connect with exactly one iADT connection, and one iADT connection shall connect with exactly one ADT port (i.e., an iADT port in one device is uniquely connected with one iADT port in another device).

The means by which one port learns the internet address and port number of a remote port in order to establish a connection is beyond the scope of this standard. The registered port number 4169 has been allocated by the IANA for the exclusive use of iADT.

One physical network port may respond to multiple internet addresses, and thus may contain multiple iADT sockets, one for each of the internet addresses. If a device (e.g., device A) contains only one iADT socket (i.e., there is one Ethernet port and it responds to a single internet address) then that iADT socket may be connected to iADT sockets in different devices. Each of those iADT connections connects to a different ADT port in device A. (See Figure 12.)

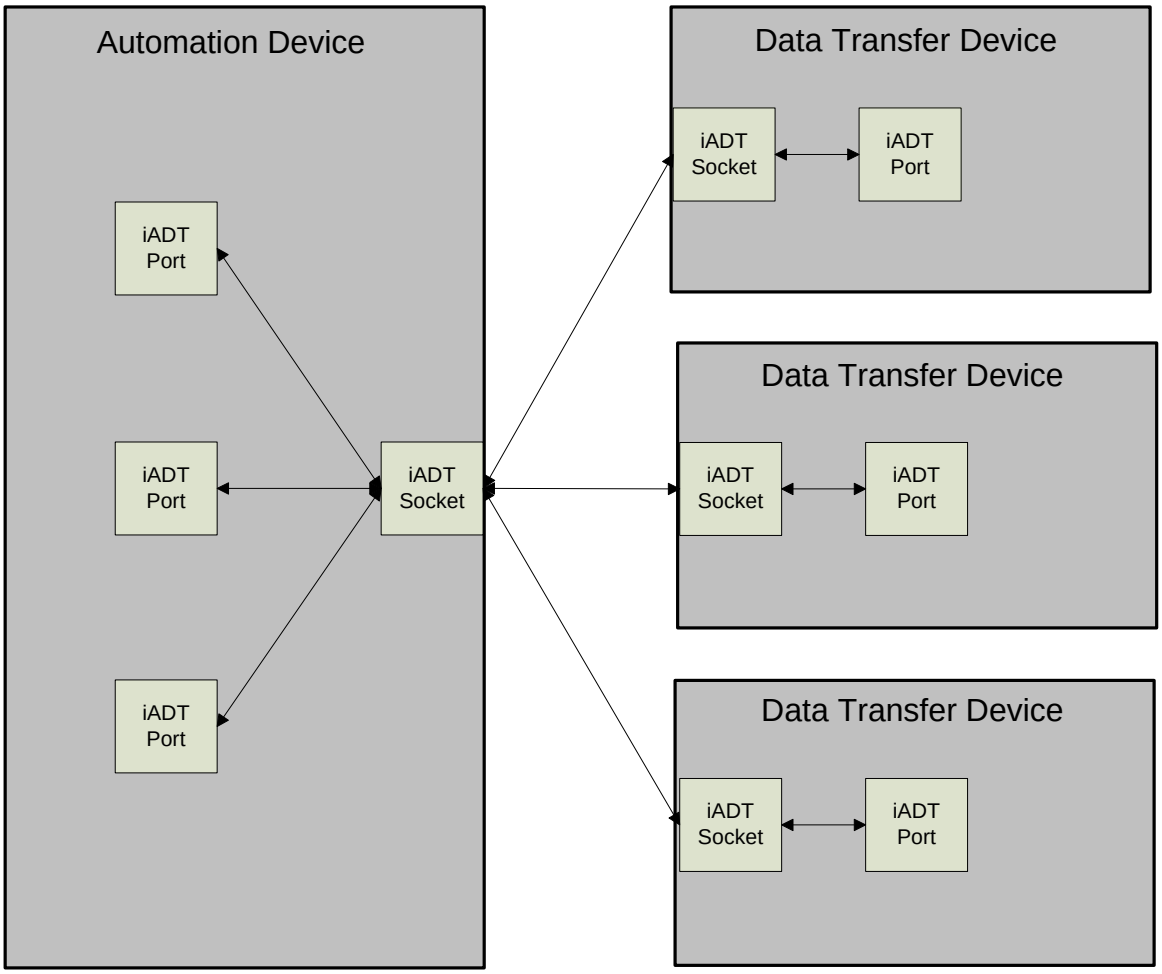

Figure 12 – Example of shared iADT socket in an automation device

Editor's Note 1: Should some of this go in the model section?

6.2 TCP layer protocol services

6.2.1 TCP layer protocol services introduction

The TCP layer protocol services consist of a number of service requests, service confirmations, and service indications to establish and remove connections, to send and receive data, and to perform other functions on the connection. RFC 793 specifies a user API for invoking TCP functions; this API consists of a number of calls. RFC 793 describes information to be passed back to the user process, but does not provide an API. This clause provides a set of service requests, service confirmations, and service indications for use by ADI ports. The mapping of service requests to TCP calls is also specified.

For the state diagram of a TCP connection, see RFC 793.

Editor's Note 2: Do we need a service indication to tell that an internet address has been assigned? Prior to that, a connection cannot be opened. This is probably an implementation detail.

6.2.2 The OPEN service request

An iADT port uses the OPEN service request to open a connection to a remote iADT port. The OPEN service request performs a TCP open call as specified in RFC 793. The arguments to the TCP open call shall be as specified in Table 9.

Service Response = **OPEN (IN (Active, [Remote Internet Address], [Timeout]), OUT ([Local Connection Name]))**

Input arguments:

Active: A Boolean value indicating whether the port shall perform an active or passive OPEN. A value of TRUE indicates that an active OPEN shall be performed. A value of FALSE indicates that a passive OPEN shall be performed.

Remote Internet Address: The internet address of the remote port. If the value of Active is TRUE, the Remote Internet Address shall be included. If the value of Active is FALSE, the Remote Internet Address shall not be included.

Timeout: The time allowed for the TCP layer to deliver data. If this time is exceeded, then the connection is closed.

Output arguments:

Local Connection Name: The identifier for the connection. The Local Connection Name argument is reported if and only if the Service Response is OK

Service Response assumes one of the following values:

OK: The call completed successfully and returned OK.

CONNECTION ILLEGAL: The call failed and returned ERROR: CONNECTION ILLEGAL FOR THIS PROCESS.

INSUFFICIENT RESOURCES: The call failed and returned ERROR: INSUFFICIENT RESOURCES.

FOREIGN SOCKET UNSPECIFIED: The call failed and returned ERROR: FOREIGN SOCKET UNSPECIFIED.

CONNECTION ALREADY EXISTS: The call failed and returned ERROR: CONNECTION ALREADY EXISTS.

Table 9 — TCP open call argument values

TCP open call argument	Value
local port	iADT port number
foreign socket	Remote Internet Address and remote iADT port number
active/passive	If Active argument of RECEIVE service request is TRUE, then 1. If Active argument of RECEIVE service request is FALSE, then 0.
timeout	Should be the global timeout value, five minutes, specified in RFC 793. This timeout should be much longer than the ADT acknowledgement time-out period (see 4.10), to allow ADT link level error recovery (see 4.6) to proceed without closing the connection.
precedence	Not specified by this standard.
security/compartment	Not specified by this standard.
options	Not specified by this standard.

6.2.3 The SEND service request

An iADT port uses the SEND service request to send data to the remote iADT port. The SEND service request performs a TCP send call as specified in RFC 793. The arguments to the TCP send call shall be as specified in Table 10.

If a subsequent SEND service request is invoked before all of the data in the buffer specified by a previous SEND service request, then all of the data in the buffer for the previous invocation shall be sent before any data in the buffer of the subsequent invocation is sent.

Service Response = **SEND (IN (Local Connection Name, Buffer, Buffer Size))**

Input arguments:

Local Connection Name: 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 bytes of encoded data to be transmitted on the connection.

Service Response assumes one of the following values:

OK: The call completed successfully and returned OK.

CONNECTION ILLEGAL: The call failed and returned ERROR: CONNECTION ILLEGAL FOR THIS PROCESS.

CONNECTION DOES NOT EXIST: The call failed and returned ERROR: CONNECTION DOES NOT EXIST.

INSUFFICIENT RESOURCES: The call failed and returned ERROR: INSUFFICIENT RESOURCES.

FOREIGN SOCKET UNSPECIFIED: The call failed and returned ERROR: FOREIGN SOCKET UNSPECIFIED.

CONNECTION CLOSING: The call failed and returned ERROR: CONNECTION CLOSING.

Table 10 – TCP send call arguments

TCP send call argument	Value
local connection name	Local Connection Name returned by TCP open call
buffer address	Buffer argument of SEND service request
byte count	Buffer Size argument of SEND service request
PUSH flag	Should be set to one when one or more end of frame (EOF) characters (see 5) are in the buffer.
URGENT flag	Zero.
timeout	Shall be the same as the timeout specified in the TCP open call that established the connection.

6.2.4 The RECEIVE service request

An iADT port uses the RECEIVE service request to receive data from a remote iADT port. The RECEIVE service request performs a TCP receive call as specified in RFC 793. The arguments to the TCP receive call shall be as specified in Table 11. The data received shall be processed as specified in clause 7.

If the RECEIVE service request is invoked a second time before enough data had been received to fill the buffer specified by the previous invocation, then the buffer specified in the first invocation shall be filled before any data is placed in the buffer indicated by the second invocation.

Service Response = **RECEIVE (IN (Local Connection Name, Buffer, Buffer Size))**

Input arguments:

Local Connection Name: The identifier for the connection.

Buffer: A buffer containing data to be transmitted. The data in the buffer shall be encoded (see 6.2).

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

Service Response assumes one of the following values:

OK: The call completed successfully and returned OK.

CONNECTION ILLEGAL: The call failed and returned ERROR: CONNECTION ILLEGAL FOR THIS PROCESS.

CONNECTION DOES NOT EXIST: The call failed and returned ERROR: CONNECTION DOES NOT EXIST.

INSUFFICIENT RESOURCES: The call failed and returned ERROR: INSUFFICIENT RESOURCES.

CONNECTION CLOSING: The call failed and returned ERROR: CONNECTION CLOSING.

Table 11 – TCP receive call arguments

TCP receive call argument	Value
local connection name	Local Connection Name returned by TCP open call
buffer address	Buffer argument of RECEIVE service request
byte count	Buffer Size argument of RECEIVE service request

6.2.5 The CLOSE service request

An iADT port uses the CLOSE service request to close a connection. The CLOSE service request performs a TCP close call as specified in RFC 793. The arguments to the TCP close call shall be as specified in Table 12.

Service Response = CLOSE (IN (Local Connection Name))

Input arguments:

Local Connection Name: The identifier for the connection.

Service Response assumes one of the following values:

OK: The close completed successfully and returned OK.

CONNECTION ILLEGAL: The call failed and returned ERROR: CONNECTION ILLEGAL FOR THIS PROCESS.

CONNECTION CLOSING: The call failed and returned ERROR: CONNECTION CLOSING.

Table 12 – TCP close call arguments

TCP send call argument	Value
local connection name	Local Connection Name returned by TCP open call

6.2.6 The STATUS service request

An iADT port uses the STATUS service request to obtain the status of a connection to another iADT port. The STATUS service request performs a TCP status call as specified in RFC 793. The arguments to the TCP status call shall be as in Table 13.

Service Response = STATUS (IN (Local Connection Name), OUT([State]))

Input arguments:

Local Connection Name: The identifier for the connection.

Output arguments:

State: The connection state returned by the TCP status call. This argument is returned if and only if the Service Response is OK.

Service Response assumes one of the following values:

OK: The call completed successfully and returned OK.

CONNECTION DOES NOT EXIST: The call failed and returned ERROR: CONNECTION DOES NOT EXIST.

Table 13 – TCP status call arguments

TCP status call argument	Value
local connection name	Local Connection Name returned by TCP open call

Editor's Note 3: Do we need this service request?

~~6.2.7 The ABORT service request~~

~~An iADT port uses the ABORT service request to abort all pending SEND and RECEIVE service requests on a connection. The ABORT service request performs a TCP abort call as specified in RFC 793. The arguments to the abort call are as specified in Table 14.~~

~~**Service Response = ABORT (IN (Local Connection Name))**~~

~~Input arguments:~~

~~**Local Connection Name:** The identifier for the connection.~~

~~**Service Response** assumes one of the following values:~~

~~**OK:** The call completed successfully and returned OK.~~

~~**CONNECTION DOES NOT EXIST:** The call failed and returned ERROR: CONNECTION DOES NOT EXIST.~~

~~**Table 14 – TCP abort call arguments**~~

~~TCP abort call argument~~	~~Value~~
~~local connection name~~	~~Local Connection Name returned by TCP open call~~

Editor's Note 4: Deleted this service request. It causes loss of the TCB, which implicitly CLOSEs the connection.

6.2.7 The OPENED service confirmation

The iADT port invokes the OPENED service confirmation to indicate that the requested connection has been opened.

OPENED (IN (Local Connection Name))

Input arguments:

Local Connection Name: The identifier for the connection.

6.2.8 The SENT service confirmation

The iADT port invokes the SENT service confirmation to indicate that a number of bytes have been sent. There is not a one-to-one correspondence between invocations of SEND and invocations of SENT. I.e., bytes sent may all be from the buffer specified by a single invocation of the SEND service request or they may span the buffers specified by the invocations of multiple SEND service requests.

SENT (IN (Local Connection Name, Sent Byte Count))

Input arguments:

Local Connection Name: The identifier for the connection.

Sent Byte Count: The number of bytes sent.

6.2.9 The RECEIVED service indication

The iADT port invokes the RECEIVED service indication to indicate that a number of bytes have been received. There is not a one-to-one correspondence between invocations of RECEIVE and invocations of RECEIVED. I.e., multiple invocations of RECEIVED may be required to indicate that the buffer specified in one invocation of RECEIVE has been filled, and a single invocation of RECEIVED may indicate that bytes have been received which span the buffers specified in multiple invocations of RECEIVE.

RECEIVED (IN (Local Connection Name, Received Byte Count))

Input arguments:

Local Connection Name: The identifier for the connection.

Received Byte Count: The number of bytes received

6.2.10 The CLOSED service indication

The iADT port invokes the CLOSED service indication when the connection has been closed. The iADT port shall not invoke either the SENT service confirmation or the RECEIVED service indication when the connection is closed. (See 6.3.2.)

CLOSED (IN (Local Connection Name))

Input arguments:

Local Connection Name: The identifier for the connection.

~~6.2.12 The ABORTED service indication~~

~~The iADT port invokes the ABORTED service indication to indicate that all pending data transmissions have been aborted. A pending data transmission is that part of the contents of one or more data buffers which have been passed to the iADT port, but for which a SENT service confirmation has not been received.~~

~~The connection remains open after ABORTED is invoked.~~

~~**ABORTED (IN (Local Connection Name))**~~

~~Input arguments:~~

~~**Local Connection Name:** The identifier for the connection.~~

Editor's Note 5: Deleted this service indication. ABORT causes loss of the TCB, which implicitly CLOSEs the connection. It would thus map to a CLOSE.

6.3 iADT port operations

6.3.1 Establishing a connection

An iADT port shall open a connection before sending any ADT information units (IUs). To open a connection between two ports, both ports invoke the OPEN service request; at least one of the invocations shall specify an Active argument with a value of TRUE. Therefore the invocations of OPEN by two iADT ports A and B can occur in three combinations which will establish a connection:

a) A specifies Active = TRUE and B specifies Active = FALSE;

b) A specifies Active = FALSE and B specifies Active = TRUE; or

c) A and B specify Active = TRUE.

If both A and B invoke OPEN with Active = FALSE, then a connection shall not be established.

When a connection has been established, the OPENED service indication shall be invoked in each port. The Local Connection Name identifies the connection. For purposes of transferring SCSI commands, data, responses, and task management requests that use the iADT port, the Local Connection Name identifies the I_T nexus. An iADT port shall not invoke the SEND, RECEIVE, CLOSE, or ABORT service requests until an OPENED service indication has been invoked.

6.3.2 Removing a connection

An iADT port invokes the CLOSE service request to remove a connection. The Port State Machine of an iADT port shall be in either P0:Initial state or P3:Logged Out state before the CLOSE service request is invoked. An iADT port is not required to invoke the CLOSE service request after the Port State Machine transitions from P2:Logged In to P3:Logged Out or from P3:Logged Out to P0:Initial.

Invocation of the CLOSED service indication indicates that the connection has been closed. If the CLOSED service indication is invoked and the Port State Machine is in the P1:Login state or the P2:Logged In state, then an I_T nexus loss event shall occur. If the CLOSED service indication is invoked, the Port State Machine shall enter P0:Initial state.

An iADT port shall not invoke the SEND, RECEIVE, CLOSE, or ABORT service requests after a CLOSED service indication has been invoked.

Editor's Note 6: Should a transition be added to the Port State Machine figure? Use prior to power down or reset.

6.3.3 Sending data

An iADT port sends an ADT information unit by invoking the SEND service request one or more times. One invocation of SEND may send more or less than one information unit.

When one or more invocations of SENT indicate that all of the data in the buffer specified by an invocation of SEND has been sent, then SEND must be invoked again to send more data.

6.3.4 Receiving data

Before an iADT port can receive data, it must invoke the RECEIVE service request to specify the buffer to receive the data. RECEIVE may be invoked multiple times before any of the specified buffers has been filled.

When an iADT port receives data from a connection, it invokes the RECEIVED service indication. One invocation of RECEIVED may indicate the arrival of less than a complete information unit or of more than one information unit. The received data shall be processed as described in 7.

When one or more invocations of RECEIVED indicate that the buffer specified by an invocation of RECEIVE has been filled, then RECEIVE shall be invoked again to provide a buffer to receive more data.

Editor's Note 7: Should counters of bytes to send and bytes received be defined and the effects of the above operations on the counters defined?

6.4 TCP connections over Ethernet

An Ethernet port (see IEEE 802.3) used for transmission of ADT frames shall support autonegotiation of link speed.

Renumbering

Clauses presently numbered 6 and higher are renumbered to 7 and higher. Notes, figures, and tables will also need to be renumbered.