**TO:** T10 Membership
**FROM:** Paul A. Suhler, Quantum Corporation
**DATE:** 21 January 2009~~14 January 2009~~
**SUBJECT:** T10/08-409r5, 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:

~~9~~• Support for short-lived connections, i.e., not having connection loss cause I_T nexus loss.
~~10~~• Specifying what to do if a connection cannot be established.
~~11~~• Relevance of the Sense$_a$ signal.
~~12~~• Removing all references to Ethernet.
~~13~~• 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.

2 Twelfth revision (11 November 2008)
Incorporated changes from the 3 November 2008 teleconference, minutes in T10/08-445r0.
Specified that a disconnect while not Paused shall cause an I_T nexus loss and implemented this by augmenting the Port and Transmitter state machines.

Removed mention of Sockets API calls in Connection state machine transitions.
Added C5:Listening and Connecting state to resolve race condition.
For reference during discussions, added a copy of the TCP state machine from RFC 793 and illustrated its mapping to the Connection state machine; see the General discussion section.
This revision still requires a cleanup of the wording of the Connection state machine to make it consistent with the descriptions of existing state machines.

3 Thirteenth revision (1 December 2008)
Incorporated changes from the 12 November 2008 teleconference, minutes in T10/08-462r0.
Moved Sockets API information to an informative annex.
Specified use of two TCP state machines.
Always listen for TCP connection on port 4169.
Included some IBM comments as notes.

4 Fourteenth revision (18 December 2008)
Incorporated changes from the 3 December 2008 teleconference, minutes in T10/09-010r0.
Modified Fig. 3 (4) to show both sADT and iADT connections.
Use “link layer” consistently rather than “session layer.”
Modified layer figures to show “transport layer” and an instance of the STPL. Reordered descriptive paragraphs.
Added definition of logout duration time and used it in the text.
Added text and Fig. 14a to clause 6 about resolving simultaneous connections, and put example ladder diagrams in a new informative Annex E.
Removed connection-related services from sADT, and added a column to Table 14 indicating whether each service is supported by iADT or both sADT and iADT.
Removed Connection state machine.
Added a statement to the first paragraph of each connection request’s description in 6.2 stating when it shall not be invoked. This replaces the table of allowed service requests for each state in the (now-removed) connection state machine.
Added a table for each service request explaining for each service response the meaning and error recovery procedure.
Finished purging references to Listen service request.
Incorporated IBM service discovery proposal.
Added numerous comments in green to highlight particular points requiring review.

5 Fifteenth revision (21 January 2008)
Incorporated changes from the 12 January 2009 Meeting, minutes in T10/08-462r0.
Added IBM’s changes to 6.2.6 AER control IU.
Added IBM’s text for implicit logout to 4.3.2.4.4 Transition P2:Logged-In to P3:Logged-Out.

## 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 minimizes 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. In 08-409r2, the session is defined to be a pair { local IP address, remote IP address }. These IP addresses shall remain constant for the lifetime of the session.
- This standard requires that a TCP server for iADT listen on port number 4169. Use of the iADT protocol to connect to other port numbers is 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 $\text{Sense}_a$ 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 $\text{Reset}_a$ connection is optional. In Ethernet, this will cause either a disconnection or a hard reset.
- Support for the $\text{Sense}_d$ connection is optional.
- 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 discussed 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. The final decision was that discovery would 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. Baud Rate is not used, but a fixed value can be specified, probably 9,600. |
| 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 | Yes – This error can occur. |
| 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. |

The result of the analysis and design was that iADT would support the entire ADT link layer, with modifications only as needed to support transient connections.

### TCP Connection State Diagram

For reference, the TCP connection state diagram is reproduced from RFC 793, figure 6:

```
                              +---------+ ---------\      active OPEN
                              |  CLOSED |            \    -----------
                              +---------+<---------\   \   create TCB
                                |     ^              \   \  snd SYN
                   passive OPEN |     |   CLOSE        \   \
                   ------------ |     | ----------       \   \
                    create TCB  |     | delete TCB         \   \
                                V     |                      \   \
                              +---------+            CLOSE    |    \
                              |  LISTEN |          ---------- |     |
                              +---------+          delete TCB |     |
                   rcv SYN      |     |     SEND              |     |
                  -----------   |     |    -------            |     V
 +---------+      snd SYN,ACK  /       \   snd SYN          +---------+
 |         |<-----------------           ------------------>|         |
 |   SYN   |                    rcv SYN                     |   SYN   |
 |   RCVD  |<-----------------------------------------------|   SENT  |
 |         |                    snd ACK                     |         |
 |         |------------------           -------------------|         |
 +---------+   rcv ACK of SYN  \       /  rcv SYN,ACK       +---------+
   |           --------------   |     |   -----------
   |                  x         |     |     snd ACK
   |                            V     V
   |  CLOSE                   +---------+
   | -------                  |  ESTAB  |
   | snd FIN                  +---------+
   |                   CLOSE    |     |    rcv FIN
   V                  -------   |     |    -------
 +---------+          snd FIN  /       \   snd ACK          +---------+
 |  FIN    |<-----------------           ------------------>|  CLOSE  |
 | WAIT-1  |------------------                              |   WAIT  |
 +---------+          rcv FIN  \                            +---------+
   | rcv ACK of FIN   -------   |                            CLOSE  |
   | --------------   snd ACK   |                           ------- |
   V        x                   V                           snd FIN V
 +---------+                  +---------+                   +---------+
 |FINWAIT-2|                  | CLOSING |                   | LAST-ACK|
 +---------+                  +---------+                   +---------+
   |                rcv ACK of FIN |                 rcv ACK of FIN |
   |  rcv FIN       -------------- |    Timeout=2MSL -------------- |
   |  -------              x       V    ------------        x       V
    \ snd ACK                 +---------+delete TCB         +---------+
     ------------------------>|TIME WAIT|------------------>| CLOSED  |
                              +---------+                   +---------+
```

**TCP Connection State Diagram**

### *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 768, *User Datagram Protocol*

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*

RFC 5246, *The Transport Layer Security (TLS) Protocol, version 1.2*

#### 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 IP:** Internet protocol (see RFC 791 and RFC 2460).

**3.1.x LLC:** Ethernet link layer control.

**3.1.x Logout duration time:** the length in seconds that a port that receives a Port Logout IU shall remain in P3:Logged-out state (see 7.5.5).

**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).

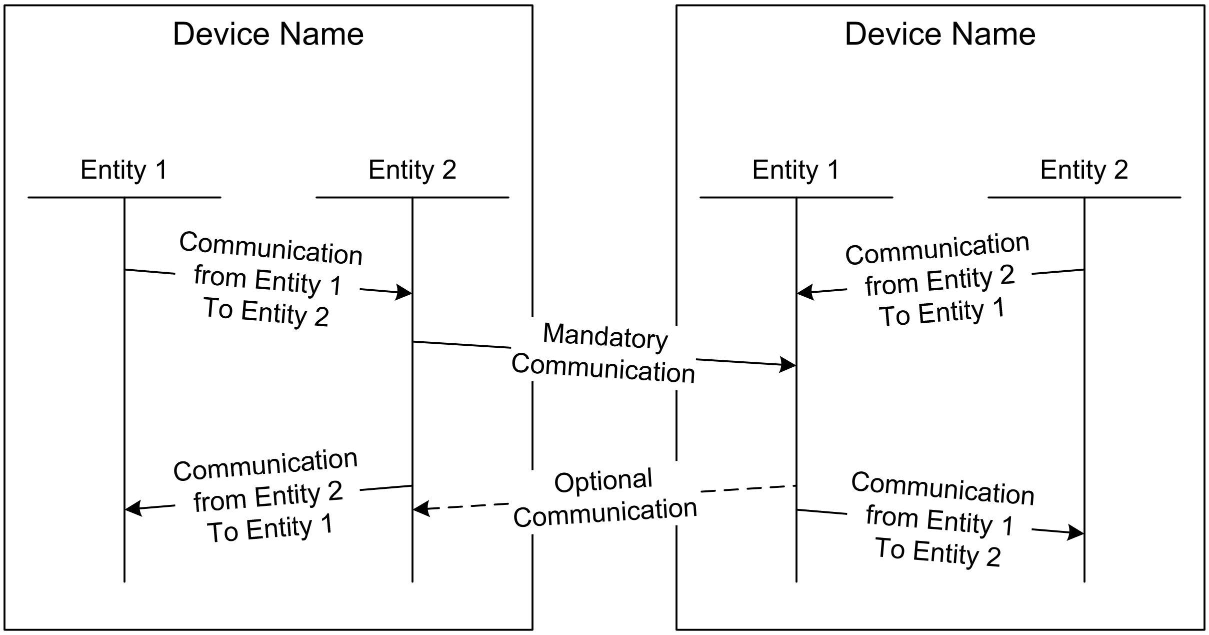

Device Name
Entity 1
Entity 2
Communication from Entity 1 To Entity 2
Communication from Entity 2 To Entity 1
Mandatory Communication
Optional Communication
Device Name
Entity 1
Entity 2
Communication from Entity 2 To Entity 1
Communication from Entity 1 To Entity 2

**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~~ examples of serial ADT (sADT) and internet ADT (iADT) interfaces (see 4.1.2) 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 may communicate~~s~~ 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.

> Note: A media changer would not necessarily use both sADT and iADT ports. They are both shown for comparison.

Replace and renumber Figure 3:

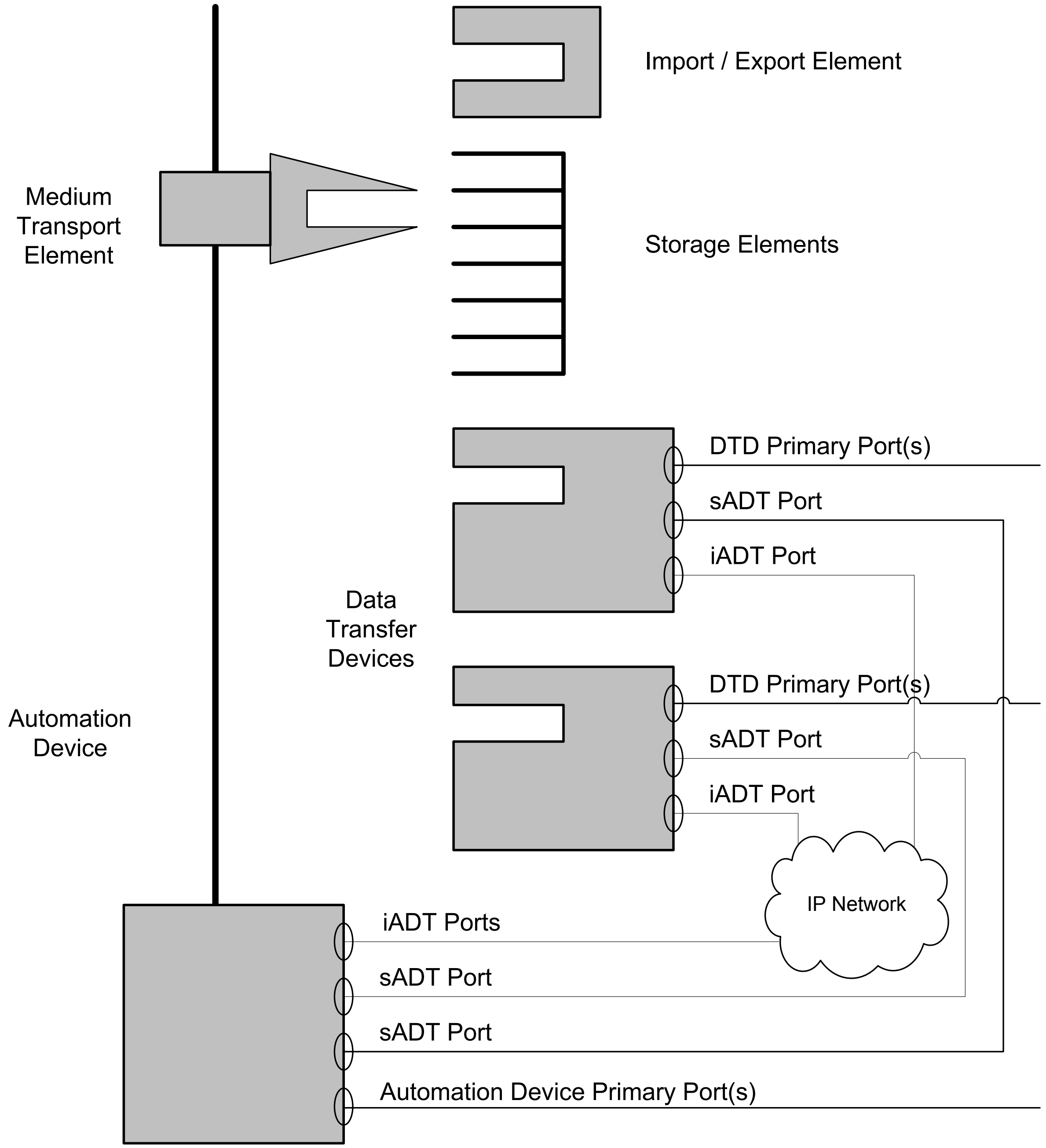


**Figure ~~3~~4 — Example Media Changer application of ADT**

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.

### 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 STPL defined by ADT consists of the ADT Transport Layer. The Interconnect Layer defined by ADT consists of three layers, the Link Layer, the Connection Layer, and the Physical Layer.

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.

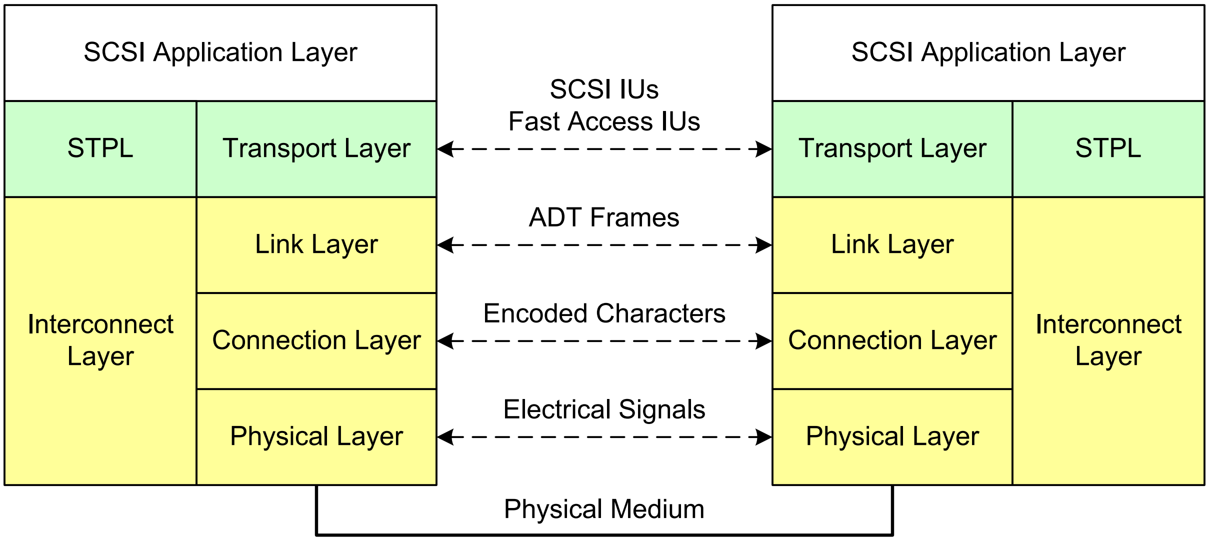


**Figure 5 – ADT communication model**

At the transport layer, information units (IUs) are passed between ADT ports. At the link 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.

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

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 ADT link layer (see clause 7) provides establishment of sessions (see 3.1.x) between pairs of ADT ports and provides reliable transmission of ADT frames between the two ADT ports~~ in a session~~.

The ADT connection layer (see clause 6) provides transmission of encoded characters between ADT ports. Two alternative transmission methods are provided by 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) and provides service discovery using UDP (see RFC 768). The TCP connection uses the Internet Protocol (IP) (see RFC 791 and RFC 2460) to provide transmission over a physical layer (e.g., Ethernet). The iADT protocol also provides transmission over a TCP connection secured using the Transport Layer Security (TLS) protocol (see RFC 5246).

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 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 link layer and the connection layer is called the connection service interface.

Figure 6 shows the serial ADT (sADT) hierarchy of protocols which may be used to implement ADT on the RS-422 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 | | Link layer |
| sADT | | Connection Layer |
| RS-422 | | Physical Layer |

**Figure 6 – sADT protocol hierarchy**

Figure 7 ~~shows~~ is an example of the Internet ADT (iADT) hierarchy of protocols using the Ethernet physical layer (see IEEE 802.3-2005 and 5.2.5.3). iADT ~~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 | | Link layer |
| iADT | | Connection Layer |
| TCP | | |
| IP | | |
| Ethernet LLC | | |
| Ethernet MAC | | |
| Ethernet PHY | | Physical Layer |

**Figure 7 – iADT protocol hierarchy**

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.

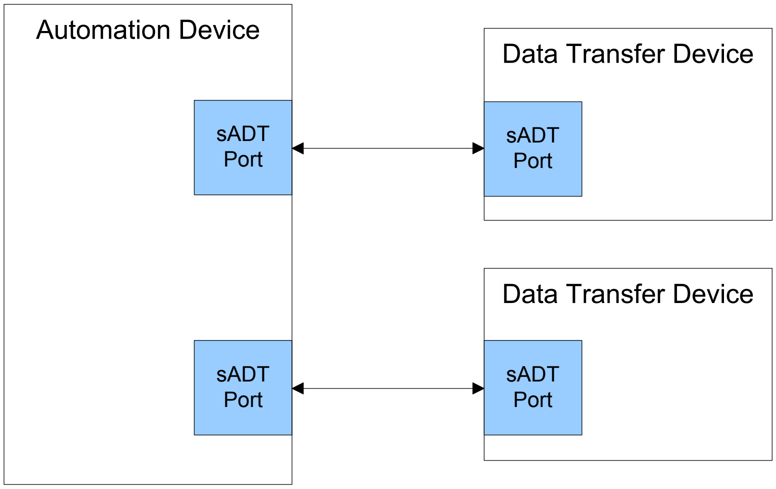


**Figure 8 – sADT port example**

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.

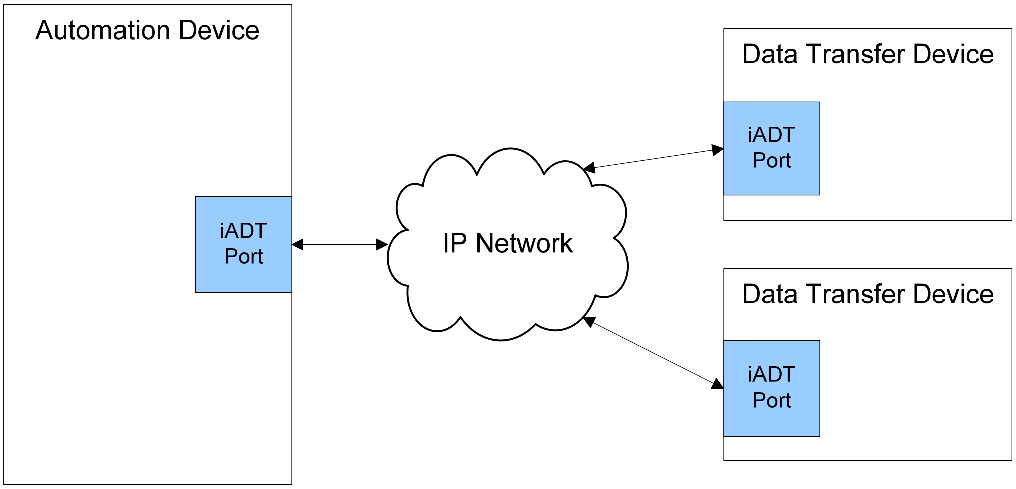


**Figure 9 – iADT port example**

## 4.2 Default operating parameters

The default operating parameters for a port are as follows:

a) the baud rate shall be set to 9 600 by an sADT port~~ and to 0 by an iADT port~~;
b) the ACK offset shall be set to 1; and
c) the Maximum Payload size shall be 256 bytes.

These values shall remain in effect until the login process is complete, at which time the negotiated values shall take effect.

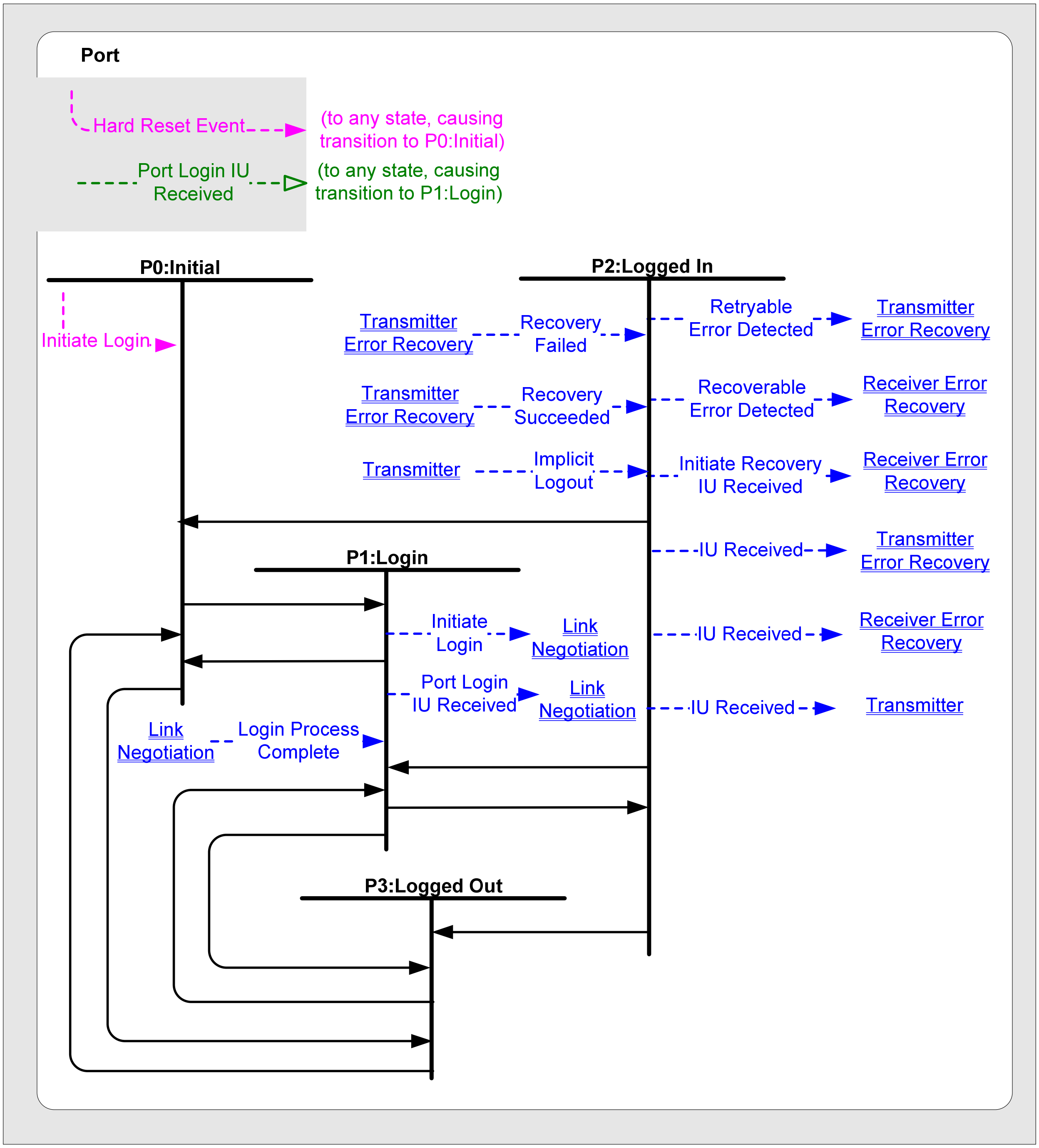
## 4.3 ADT state machines

<…>

### 4.3.2 Port state machine

<…>

Add the Implicit Logout message to P2:Logged-In state in the Port state machine figure:



<…>

No changes to 4.3.2.4:

### ~~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 between the two ADT ports which performed link negotiation. A session is uniquely defined for sADT by the two physically-connected sADT ports and for iADT by the IP addresses of the two logged-in iADT ports.~~

~~Note: Is the above definition of session adequate? Does that level of detail belong in 3.1.x?~~

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

~~<…>~~

#### 4.3.2.4.4 Transition P2:Logged-In to P3:Logged-Out

A port shall transition to P3:Logged-Out state after it receives a Port Logout IU and sends the corresponding ACK IU.

A port shall transition to P3:Logged-Out state and set the logout duration time ~~to zero or~~ to a vendor-specific value which may be zero after it receives an Implicit Logout message.

When a port in a DT device receives an Implicit Logout message it shall:

a) abort all open exchanges;
b) disable Asynchronous Event Reporting; and
c) set port operating parameters to default.

When a port in an automation device receives an Implicit Logout message it shall:

a) abort all open exchanges; and
b) set port operating parameters to default.

<…>

#### 4.3.2.5.2 Transition P3:Logged-out to P0:Initial

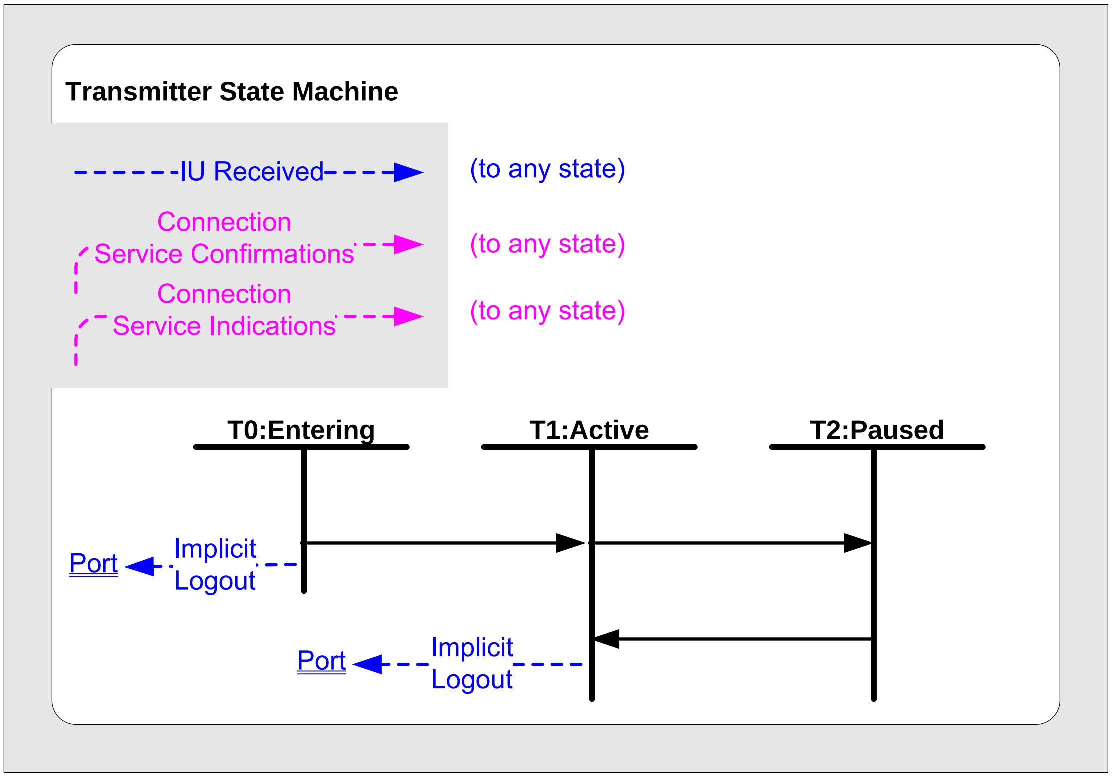
A port shall transition to P0:Initial after the logout duration time ~~specified in the Port Logout IU (see 6.5.5)~~ has expired.

<…>

### 4.3.4 Transmitter state machine

<…>

Add the Implicit Logout message to the Transmitter state machine figure and delete the unused transition from T1:Active to T0:Entering:



<…>

### 4.3.4.2 T0:Entering state

#### 4.3.4.2.1 State description

On entry to this state the sADT port shall start a 100 millisecond timer. On entry to this state the iADT port may or may not start a 100 millisecond timer.

While in this state, a port shall not transmit ADT information units.

~~Note: Added "information units" to avoid letting implementers think that iADT ports could not transmit Ethernet packets containing data other than ADT IUs.~~

If a **Disconnected** service confirmation or a ~~**Disconnect received**~~**Disconnect Received** service indication is invoked by an iADT port while in this state, the port shall send an Implicit Logout message to the port state machine.

<…>

### 4.3.4.3 T1:Active state

#### 4.3.4.3.1 State description

A port in T1:Active state may transmit and receive all types of information units.

If a **Disconnected** service confirmation or a ~~**Disconnect received**~~**Disconnect Received** service indication is invoked by an iADT port while in this state, the port shall send an Implicit Logout message to the port state machine.

<…>

### 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) is an sADT port and detects the change of state of the Sense line from presence to absence (i.e., Sense$_a$ for DT device port and ~~Sense$_b$~~ Sense$_d$ for automation device port (see figure 11); ~~or~~
~~e) detects the assertion of the Reset$_a$ line (see table 13).~~
e) receives a ~~Reset~~**Reset Received** service indication (see 6.6.10); or
f) receives an Implicit Logout message while the Port state machine is in the P2:Logged-In state.

<…>

### 4.10.1 Acknowledgement time-out period calculation

When changing operating parameters (see 3.1.32), ~~a~~ 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 (Acknowledgement time-out period) ~~to 15~~ and rename to “sADT port acknowledgement time-out period.” Add the following paragraph immediately following the figure:

An iADT port shall use an initial acknowledgement time-out period of 2,500 milliseconds. This may be changed if the iADT 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 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~~ may be 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)~~a) the output is an open-collector type;
~~b)~~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 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 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~~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 sADT 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 sADT ports.

### 5.3.4 Ethernet transmit-receive connection instances

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

**Table 11 – Ethernet transmit-receive connections for ~~Ethernet~~ iADT ports using Ethernet**

| 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 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 Ethernet LED connection instances

Table 12 defines the LED connections used by the DT device supporting Ethernet transmit-receive connections.

**Table 12 – Ethernet LED connections**

| Connection Name | O/M[a] | Connection Type | ~~Driven~~ Asserted 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~~ sADT 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. A connection shall exist implicitly between two sADT ports which are physically connected. A connection between two iADT ports ~~is~~ shall be established explicitly ~~established~~.

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

**Table 14 – ADT connection services**

| Connection service | Connection service type | Invoked by device type | Supported by port type |
|---|---|---|---|
| Connect | Request | Either | iADT |
| Connected | Confirmation | Either | iADT |
| Connect Received | Indication | Either | iADT |
| Send | Request | Either | sADT and iADT |
| Sent | Confirmation | Either | sADT and iADT |
| Receive | Request | Either | sADT and iADT |
| Received | Confirmation | Either | sADT and iADT |
| Disconnect | Request | Either | iADT |
| Disconnected | Confirmation | Either | iADT |
| Disconnect Immediate | Request | Either | iADT |
| ~~Disconnect received~~Disconnect Received | Indication | Either | iADT |
| Reset | Request | Automation device | sADT and iADT |
| ~~Reset received~~Reset Received | Indication | DT device | sADT and iADT |

~~Note: Please review the wording here requiring iADT ports to listen (“await a connection.”) Is it correct? Is it strong enough? The reference to the TCP passive OPEN is here because Table D.2 no longer has a row mapping the deleted **Listen** service request to the passive OPEN.~~

~~Note: Given that a session will have not more than one connection at a time, I’ve removed the Connection argument from the connection services. I’d originally conceived of the session argument as indicating the link layer’s opaque handle and the connection argument indicating the connection layer’s opaque handle.~~

~~Now the concept is that the link layer will assign a session handle before it first invokes **Connect** for a session, and will then use that handle for **Send**, **Receive**, and **Disconnect**. Similarly, the service responses and indications will use the session handle.~~

~~Comments?~~

~~Connection services for creating and terminating connections shall be supported by iADT ports and shall not be supported by sADT ports.~~ An iADT port in a DTD shall ~~await~~ listen for a connection from any iADT port and an iADT port in an automation device may listen for a connection from any iADT port. An iADT port may initiate a connection to a specific iADT port. An iADT port awaits a connection by performing a TCP passive OPEN specifying the iADT port number (4169) or the iADT over TLS port number <<TBD TLS>> for the local port. An iADT port initiates a connection by invoking the **Connect** service request. When the connection is established, both iADT ports receive a **Connected** service confirmation. The iADT ports may exchange TCP control segments in order to establish the connection.

All ADT IUs using the iADT over TLS port number shall use the Transport Layer Security (TLS) protocol (see RFC 5246).

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.

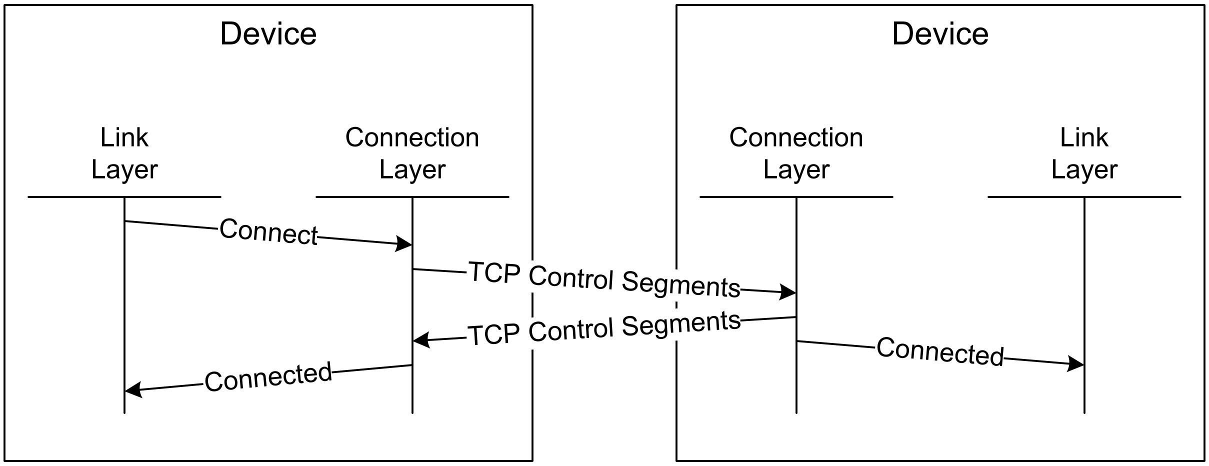


**Figure 18a – Connection services for establishing a connection between iADT ports**

~~If a connection is established in a session and a new connection is established in the same session, then the iADT port:~~

~~a)shall not invoke the **Connected** service indication for the new connection;~~
~~b)shall close one connection as specified in Table 14a; and~~

~~c)shall not invoke the **Disconnected** service confirmation or the **Disconnect received**~~**Disconnect Received**~~ service indication for the closed connection.~~

**~~Table 14a – Resolution of redundant connections in a session~~**

| ~~**Device initiating first connection established**~~ | ~~**Device initiating second connection established**~~ | ~~**Action to resolve conflict**~~ |
|---|---|---|
| ~~Automation device~~ | ~~Automation device~~ | ~~DTD closes connection established first~~ |
| ~~Automation device~~ | ~~DTD~~ | ~~DTD closes connection it initiated~~ |
| ~~DTD~~ | ~~Automation device~~ | ~~DTD closes connection it initiated~~ |
| ~~DTD~~ | ~~DTD~~ | ~~Automation device closes connection established first~~ |

~~Annex E contains examples of resolving redundant connections.~~

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

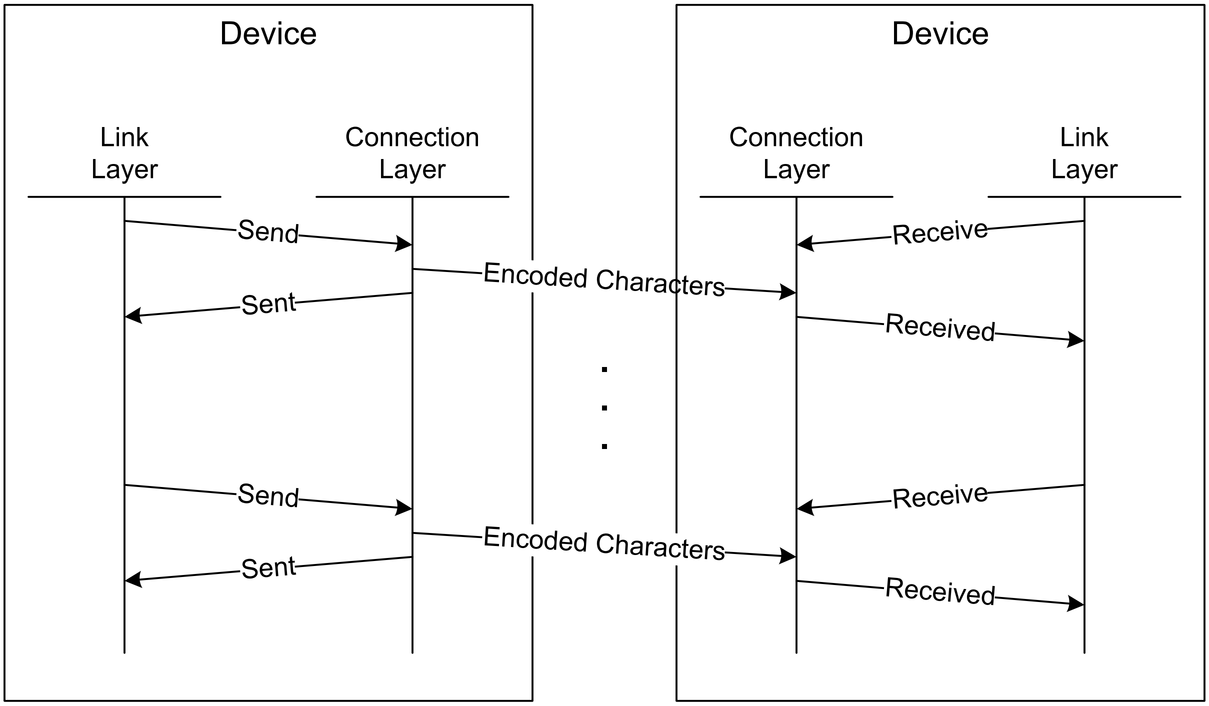


**Figure 19 – Connection services for transferring data**

An ADT port sends encoded characters on a connection within a session by invoking the **Send** service request. ~~The **Send** service request specifies the session, 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 connection 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.

~~Note: I added the following paragraph after I added the "TCP control segments" labels to the inter-port communications in figures 18a and 20a, in case there were questions about the communications in Figure 19. Is this necessary?~~

sADT ports transmit encoded characters on the RS-422 physical layer. iADT ports transmit encoded characters in TCP segments over the particular physical layer used by the iADT port.

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.

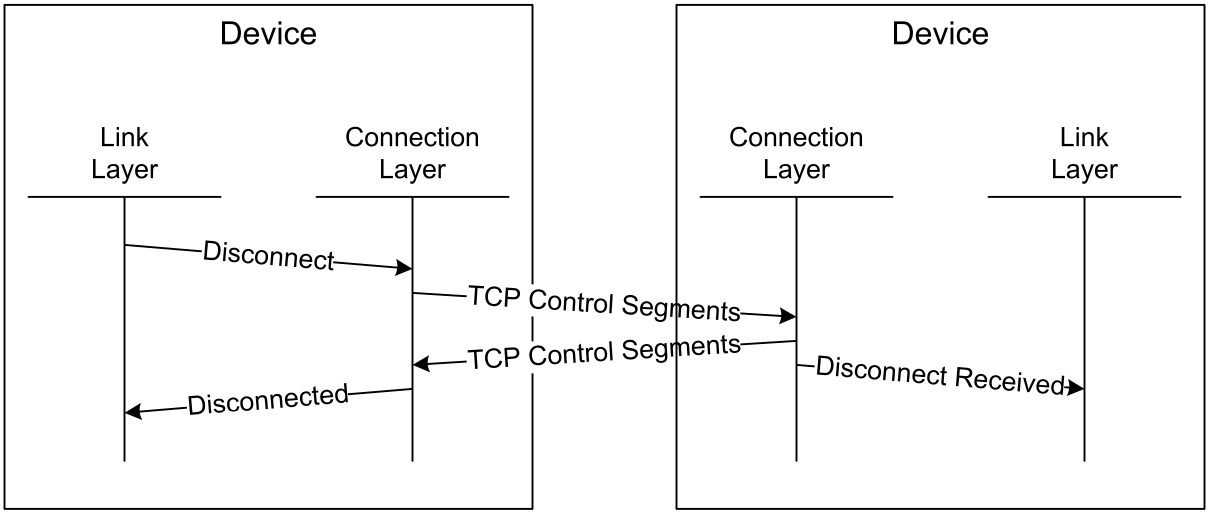


**Figure 20a – Connection services for closing a connection between iADT ports**

An iADT port closes a connection by invoking the **Disconnect** service request. An iADT port shall attempt to transmit any ~~Any~~ characters that have been successfully submitted for delivery by earlier **Send** service requests ~~will be transmitted~~ before the connection is closed. When an iADT port receives a **Disconnected** service confirmation or a ~~**Disconnect received**~~**Disconnect Received** service indication, the connection is closed and no more characters shall be received. iADT ports exchange TCP control segments in order to close the connection.

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

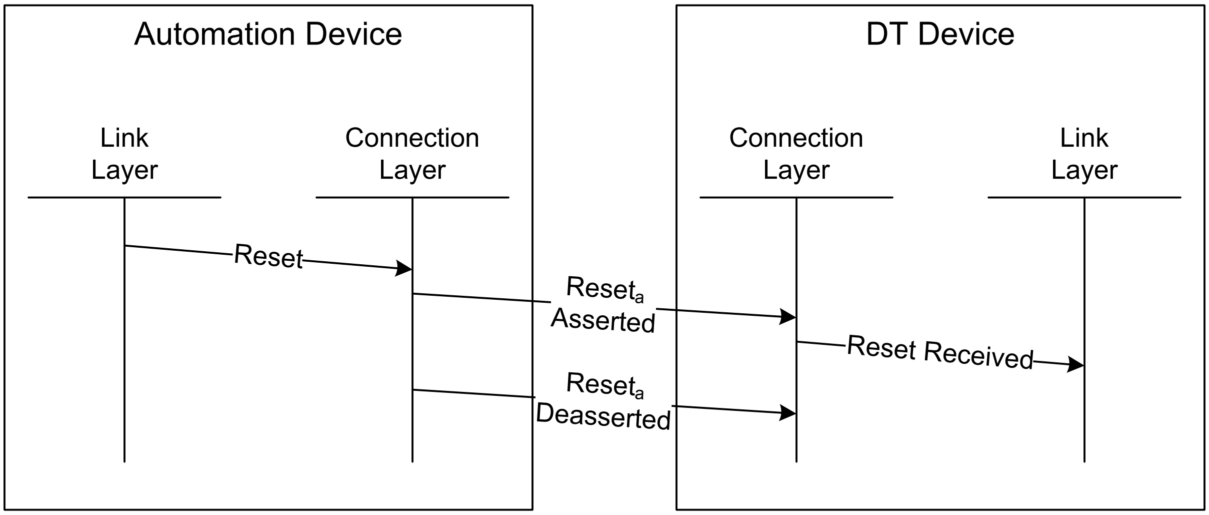


**Figure 21 - Connection services for performing a reset**

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 $\text{Reset}_a$ connection. Assertion of the $\text{Reset}_a$ connection causes the ADT port in the DT device to receive a ~~**Reset received**~~**Reset Received** service indication.

## 6.2 Connection layer connection service definitions

### 6.2.1 Connect request

A local iADT port ~~uses~~ link layer invokes the **Connect** connection service request to initiate a connection with a specific remote iADT port. ~~An iADT port shall not invoke the **Connect** service request for a session in which:~~

~~a)a connection has been established;~~
~~b)a connection has been initiated, i.e., the **Connect** service request has been invoked but neither the **Connected** service confirmation nor the **Disconnected** service confirmation has been invoked; or~~
~~c)~~a)~~a connection close has been initiated, i.e., the **Disconnect** service request has been invoked but the **Disconnected** service confirmation has not been invoked.~~

**Service Response = Connect (IN (Session Identifier, ~~Local Port, Remote Port~~Local IP Address, Remote IP Address, Method))**

Input arguments:

**Session Identifier:** ~~An~~ A link layer identifier for the session with which the connection will be associated.

~~**Local Port:** An identifier for the local port.~~

~~**Remote Port:** The identifier for the remote port.~~

**Local IP Address:** The IP address of the local iADT port.

**Remote IP Address:** The IP address of the remote iADT port.

**Method:** See Table w-1.

**Table w-1 – Method**

| Value | Description |
|---|---|

| | |
|---|---|
| Unsecured | The connection is made via unsecured TCP |
| TLS | The connection is made via Transport Layer Security (TLS). |

Note: The TCP port number of the remote port is defined in x.x.x.

**Service Response** assumes one of the values specified in Table w.

**Table w – Connect service request error processing**

| Service response | Cause | Error recovery procedure |
|---|---|---|
| GOOD | The request completed successfully | n/a |
| ~~INVALID CONNECTION STATE~~ | ~~The **Session** argument specified a session for which a connection was already established, a connection had been initiated, or a connection close had been initiated.~~ | ~~Create new connection and retry operation~~ |
| INVALID LOCAL ~~PORT~~IP ADDRESS | The **Local ~~Port~~ IP Address** argument did not specify a valid local ~~port~~IP address. | Retry request with valid local ~~port~~IP address. |
| ~~LOCAL PORT IN USE~~ | ~~The **Local Port** argument specified a local port that was unable to support any more connections.~~ | ~~Close other connection and retry request.~~ |
| INVALID REMOTE ~~PORT~~IP ADDRESS | The **Remote ~~Port~~ IP Address** argument did not specify a valid remote ~~port~~IP address. | Retry request with valid remote ~~port~~IP address. |
| NO PHYSICAL CONNECTION | The Sense connection was not asserted or because the Ethernet iADT port did not detect a signal. | Not specified by this standard |

~~Note: Do we need INVALID LOCAL PORT, LOCAL PORT IN USE, and INVALID REMOTE PORT? These would be caused by implementation errors.~~

### 6.2.2 Connected service confirmation

~~The~~ A local iADT port connection layer invokes the **Connected** service confirmation ~~notifies~~ to notify the iADT port link layer that ~~a~~ the connection initiated by a **Connect** service request has been established.

**Connected (IN (Session, Connection, Remote ~~Port~~IP Address))**

Input arguments:

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

**Connection:** An identifier for the connection.

**Remote IP Address:** The IP address of the remote iADT port.

~~**Remote Port:** The identifier for the remote port.~~

~~Note: There is a pedagogical problem with using **Connected** to indicate establishment of a connection by a remote port in the absence of a **Listen**. If this is the first connection prior to login – i.e., before the session is created – then the value of the session handle was created implicitly when the iADT port did its passive OPEN. However, from the point of view of someone only looking at the connection services, it ***looks like*** the session handle was created by the connection layer, while in all other cases it is obviously created by the link layer. Is this a potential source of confusion for implementers?~~

### 6.2.3 Connect Received service indication

A local iADT port connection layer invokes the **Connect Received** service indication to notify the iADT port link layer that a connection initiated by a remote iADT port has been established.

**Connect Received (IN (Connection, Remote IP address))**

Input arguments:

**Connection:** An identifier for the connection which was established.

**Remote IP Address:** The identifier for the remote port.

### 6.2.4~~3~~ Send service request

An ADT port ~~uses~~ link layer invokes the **Send** service request to send data on a connection. An ADT port link layer shall not invoke the **Send** service request for a session for which:

a) there is no established connection; or
b) a connection close has been initiated, i.e., the **Disconnect** service request has been invoked but the **Disconnected** service confirmation has not been invoked.

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 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 **GOOD**, 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 **GOOD**, then the characters may or may not have been transmitted on the physical 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:** An 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 values specified in Table w+1.

**Table w+1 – Send service request error processing**

| Service response | Cause | Error recovery procedure |
|---|---|---|
| **GOOD** | The request completed successfully | n/a |
| **INVALID CONNECTION STATE** | The **Session** argument did not specify a session with an established connection. | Create new connection and retry operation |
| **INVALID BUFFER** | The **Buffer** argument did not specify a valid buffer. | Retry request with valid buffer. |
| **OUT OF RESOURCES** | The port lacked resources to accept more characters for transmission. | Retry send after a delay |

### 6.2.45 Sent service confirmation

~~The~~ An ADT port connection layer invokes the **Sent** service confirmation ~~notifies~~ to notify the ADT port link layer 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 layer.

**Sent (IN (Session, Connection))**

Input arguments:

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

**Connection:** An identifier for the connection.

### 6.2.~~5~~6 Receive service request

An ADT port link layer invokes the **Receive** service request to receive data from a connection. The data received shall be processed as specified in clause 7. An ADT port link layer shall not invoke the **Receive** service request for a session for which there is no established connection.

An ~~iADT~~ ADT port link layer may invoke the **Receive** service request for a connection for which a connection close has been initiated~~,~~ (i.e., the **Disconnect** service request has been invoked but the **Disconnected** service confirmation has not been invoked).

If the **Receive** service request is invoked a second time before the **Received** service confirmation has been invoked, then the connection layer shall reject 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:** An 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 values specified in Table w+2.

**Table w+2 – Receive service request error processing**

| Service response | Cause | Error recovery procedure |
|---|---|---|
| GOOD | The request completed successfully | n/a |
| INVALID CONNECTION STATE | The request failed because the ~~**Session**~~ **Connection** argument did not specify a ~~session with an established~~valid connection. | ~~Create new connection and retry operation~~Retry operation with a valid connection |
| ~~INVALID BUFFER~~ | ~~The request failed because the **Buffer** argument did not specify a valid buffer.~~ | ~~Retry request with valid buffer~~ |
| RECEIVE PENDING | The request failed because the ADT port has invoked the **Receive** service request and the port has not yet invoked the **Received** service confirmation. | Retry **Receive** service request after processing **Received** service confirmation |

### 6.2.~~6~~7 Received service confirmation

~~The~~ A local ADT port connection layer invokes the **Received** service confirmation ~~notifies~~ to notify the ADT port link layer 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 frame may be transferred to the link layer in multiple invocations of the **Received** service indication. The beginning of the frame may not align with the start of the buffer and the end of the frame may not align with the end of the buffer (i.e., the first character in the buffer may not be an SOF and the last character in the buffer may not be an EOF).

If a connection is closed then the **Received** service confirmation may be invoked to deliver the characters which have been received. The received character count may be zero.

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

Input arguments:

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

**Connection:** An identifier for the connection.

**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. The number of characters may be less than the buffer size specified in the corresponding invocation of the **Receive** service request (e.g., in a non-blocking implementation).

### 6.2.~~7~~8 Disconnect service request

~~An~~ A local iADT port ~~invokes~~ connection layer receives the **Disconnect** service request to close a connection to a remote iADT port. An iADT port link layer shall not invoke the **Disconnect** service request for a session for which:

a) there is no established connection; or
b) a connection close has been initiated, i.e., the **Disconnect** service request has been invoked but the **Disconnected** service confirmation has not been invoked.

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

Input arguments:

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

**Connection:** An identifier for the connection.

**Service Response** assumes one of the values specified in Table w+3.

**Table w+3 – Disconnect service request error processing**

| Service response | Cause | Error recovery procedure |
|---|---|---|
| GOOD | The request completed successfully | n/a |
| INVALID CONNECTION STATE | The request failed because the **Session** argument did not specify a session with an established connection. | Not specified by this standard |

### 6.2.9 Disconnect Immediate service request

An iADT port link layer invokes the **Disconnect Immediate** service request to close a connection without receiving any further data on the connection and without invoking a service confirmation.

**Disconnect Immediate (IN (Connection))**

Input arguments:

**Connection:** An identifier for the connection.

### 6.2.~~8~~10 Disconnected service confirmation

A local iADT port connection layer invokes the ~~The~~ **Disconnected** service confirmation ~~notifies~~ to notify the iADT port link layer that either the connection has been closed or a **Connect** service request has failed. The iADT port connection layer shall not invoke the **Disconnected** service confirmation or the ~~**Disconnect received**~~**Disconnect Received** service indication until all received characters have been transferred to the iADT port link layer by use of the **Received** service confirmation.

The value in the ~~Session~~ **Connection** argument shall not be used in any service requests until it is reported in the Connection ~~Session~~ argument of a subsequent **Connected** service confirmation or in the Connection argument of a subsequent **Connect Received** service indication.

~~If an iADT port in a DT device detects the transition of the Sense$_a$ connection from asserted to deasserted, it may invoke the **Disconnect received**~~**Disconnect Received**~~ service indication. If an iADT port in an automation device detects the transition of the Sense$_d$ connection from asserted to deasserted, it may invoke the **Disconnect received**~~**Disconnect Received**~~ service indication. If an Ethernet iADT port detects loss of signal, it shall invoke the **Disconnect received**~~**Disconnect Received**~~ service indication.~~

**Disconnected (IN (Session, Connection, Reason))**

Input arguments:

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

**Connection:** An identifier for the connection.

**Reason:** The reason that the connection was closed.

**Reason** assumes one of the following values:

**DISCONNECT REQUESTED:** The iADT port processed a **Disconnect** service request.

**SENSE DEASSERTED:** The port detected transition of the Sense$_a$ connection from asserted to deasserted.

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

**CONNECTION REFUSED:** The remote iADT port rejected the connection attempt.

### 6.2.9 ~~Disconnect received~~Disconnect Received service indication

A local iADT port connection layer invokes the~~The~~ **~~Disconnect received~~Disconnect Received** service indication ~~notifies~~ to notify the iADT port link layer that the connection has been closed by the remote port. The iADT port connection layer shall not invoke the **~~Disconnect received~~Disconnect Received** service indication until all received characters have been transferred to the link layer.

The value in the ~~Session~~ Connection argument shall not be used in any service requests until it is reported in the ~~Session~~ Connection argument of a subsequent **Connected** service confirmation or in the **Connection** argument of a subsequent **Connect Received** service indication.

~~If an iADT port in a DT device detects the transition of the Sense$_a$ connection from asserted to deasserted, it may invoke the~~ **~~Disconnect received~~Disconnect Received** ~~service indication. If an iADT port in an automation device detects the transition of the Sense$_d$ connection from asserted to deasserted, it may invoke the~~ **~~Disconnect received~~Disconnect Received** ~~service indication.~~ If an ~~Ethernet~~ iADT port detects loss of signal, it ~~shall~~ may invoke the **~~Disconnect received~~Disconnect Received** service indication.

**~~Disconnect received~~Disconnect Received (IN (Session, Connection, Reason))**

Input arguments:

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

**Connection:** An identifier for the connection.

**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 port detected transition of the Sense$_a$ connection from asserted to deasserted.

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

**CONNECTION LOST:** The connection is no longer valid.

### 6.2.10 Reset service request

An ADT port link layer in an automation device ~~uses~~ invokes the **Reset** service request to reset the ADT port and assert the Reset$_a$ connection (see table 9).

**Reset (IN (~~Local Port, Remote Port~~DTD Identifier))**

~~**Local Port:** An identifier for the local port.~~

~~**Remote Port:** The identifier for the remote port.~~

**DTD Identifier:** The identifier for the DTD to be reset.

### 6.2.11 ~~Reset received~~Reset Received service indication

~~The~~ An ADT port connection layer invokes the ~~**Reset received**~~**Reset Received** service indication in a DT device ~~indicates~~ to indicate to the link layer that the ADT port has been reset by assertion of the Reset$_a$ connection (see table 9).

**~~Reset received~~Reset Received (IN (~~Local Port~~Reset$_a$ Connection Identifier))**

**~~Local Port~~Reset$_a$ Connection Identifier:** An identifier for the ~~local port~~Reset$_a$ connection that was asserted.

### ~~6.2.12 Error recovery~~

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

## 6.3 sADT port support of connection services

### 6.3.1 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 | sADT port implementation |
|---|---|
| **Session** | The identifier for the session with which the connection is associated |
| **Connection** | The identifier assigned by the connection layer for the connection. |
| **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 |
| **Connection** | The identifier assigned by the connection layer for the connection. |

### 6.3.2 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 | sADT port implementation |
|---|---|
| **Session** | The identifier for the session with which the connection is associated |
| **Connection** | The identifier assigned by the connection layer for the connection. |
| **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 | sADT port implementation |
|---|---|
| **Session** | The identifier for the session with which the connection is associated |
| **Connection** | The identifier assigned by the connection layer for the connection. |
| **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 |

### 6.3.3 Performing a reset

An automation device invokes 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 | sADT port implementation |
|---|---|
| ~~**Local Port**~~ | ~~Used to select the port in the automation device to transmit the reset to the DT device~~ |
| ~~**Remote Port**~~**DTD Identifier** | ~~Identifier for the remote port receiving the reset.~~The identifier for the DTD to be reset. |

A DT device shall treat the invocation of the ~~**Reset received**~~**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**~~**Reset Received** service indication is set by the sADT port.

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

| Argument | sADT port implementation |
|---|---|
| ~~**Local Port**~~**Reset$_a$ Connection Identifier** | ~~Indicates the DT device port which received the reset from the automation device port~~An identifier for the Reset$_a$ connection that was asserted. |

## 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 and remote IP address 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 iADT port number ~~(4169)~~ and the remote IP address associated with the session. The local iADT port may learn the IP address of the remote iADT port by service discovery using UDP (see 6.5).

The iADT port may support more than one session. There shall be not more than one session between any two iADT ports.

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.

**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~~IP Address** | The IP address of the **local port** argument to the **OPEN** call |
| **Remote ~~Port~~IP Address** | The IP address component of the **foreign socket** argument to the **OPEN** call. The port number component of the **foreign socket** argument to the **OPEN** call shall be ~~4169~~the iADT 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** service request |
| **Connection** | The identifier assigned by the connection layer for the connection. |
| **Remote ~~Port~~IP Address** | IP address of the remote port |

Table y+3 shows how the arguments to the **Connect Received** service confirmation are set by the iADT port.

**Table y+3 – Connect Received service confirmation usage by iADT port**

| Argument | iADT port implementation |
|---|---|
| **Session** | The value of the Session argument to the **Connect** service request |
| **Connection** | The identifier assigned by the connection layer for the connection. |
| **Remote IP Address** | 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 an indication that the data should be transmitted immediately~~the **PUSH flag** argument set~~. Table y+~~3~~ 4 shows how the arguments to the **Send** service request are used by the iADT port.

**Table y+~~3~~ 4 – 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 identifier assigned by the connection layer for the connection. |
| **Buffer** | **buffer address** argument to **SEND** call |
| **Buffer Size** | **byte count** argument to **SEND** call |

When the **Send** service request is successfully processed ~~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~~ 5 shows how the argument to the **Sent** service confirmation is used by the iADT port.

**Table y+~~4~~ 5 – Sent service confirmation usage by iADT port**

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

| | |
|---|---|
| **Connection** | The identifier assigned by the connection layer for the connection. |

### 6.4.3 Data reception

Table y+~~5~~ 6 shows how the arguments to the **Receive** service request are used by the iADT port.

**Table y+~~5~~ 6 – 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 identifier assigned by the connection layer for the connection. |
| **Buffer** | **buffer address** argument to **RECEIVE** call (see RFC 793) |
| **Buffer Size** | **byte count** argument to **RECEIVE** call |

Table y+~~6~~ 7 shows how the arguments to the **Received** service confirmation are used by the iADT port.

**Table y+~~6~~ 7 – Received service confirmation usage by iADT port**

| Argument | iADT port implementation |
|---|---|
| **Session** | The identifier for the session with which the connection is associated |
| **Connection** | The identifier assigned by the connection layer for the connection. |
| **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 iADT 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+8~~7~~ shows how the argument to the **Disconnect** service request is used by the iADT port.

**Table y+8~~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 identifier assigned by the connection layer for the connection. |

Table y+9 shows how the argument to the **Disconnect** service request is used by the iADT port.

**Table y+9 – Disconnect Immediate service request usage by iADT port**

| Argument | iADT port implementation |
|---|---|
| **Connection** | The identifier assigned by the connection layer for the connection. |

When an iADT port that had invoked the **Disconnect** service request enters the TCP CLOSED state (see RFC 793), it shall invoke the **Disconnected** service confirmation. Table y+~~8~~ 10 shows how the argument to the **Disconnected** service confirmation is set by the iADT port.

**Table y+~~8~~ 10 – Disconnected service confirmation usage by iADT port**

| Argument | iADT port implementation |
|---|---|
| **Session** | The identifier for the session with which the connection was associated |
| **Connection** | The identifier assigned by the connection layer for the connection. |
| **Reason** | Either **DISCONNECT REQUESTED , SENSE DEASSERTED** or **LOSS OF SIGNAL** |

When an iADT port that had not invoked the **Disconnect** service request enters the TCP CLOSED state (see RFC 793), it shall invoke the **~~Disconnect received~~Disconnect Received** service indication. Table y+~~9~~ 11 shows how the argument to the **~~Disconnect received~~Disconnect Received** service indication is set by the iADT port.

**Table y+~~9~~ 11 – ~~Disconnect received~~Disconnect Received service indication usage by iADT port**

| Argument | iADT port implementation |
|---|---|
| **Session** | The identifier for the session with which the connection was associated |
| **Connection** | The identifier assigned by the connection layer for the connection |
| **Reason** | Either **CLOSED STATE, SENSE DEASSERTED** or **LOSS OF SIGNAL** |

### 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~~ 12 shows how the argument to the **Reset** service request is used by the iADT port.

**Table y+~~10~~ 12 – Reset service request usage by iADT port**

| Argument | iADT implementation |
|---|---|
| **~~Local Port~~** | ~~Used to select the port in the automation device to transmit the reset to the DT device~~ |
| **~~Remote Port~~DTD Identifer** | ~~Identifier for the remote port receiving the reset.~~The identifier for the DTD to be reset |

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

a) as a **~~Disconnect received~~Disconnect Received** service indication (see 6.2.8) and may open a new connection; or

b) as a hard reset (see 4.7).

Table y+~~11~~ 13 shows how the argument to the **~~Reset received~~Reset Received** service indication is set by the iADT port.

**Table y+~~11~~ 13 – ~~Reset received~~Reset Received service indication usage by iADT port**

| Argument | iADT implementation |
|---|---|
| **~~Local Port~~Reset$_{a}$ Connection Identifier** | ~~Indicates the DT device port which received the reset from the automation device port~~An identifier for the Reset$_{a}$ connection that was asserted |

## 6.5 Service discovery for iADT ports

### 6.5.1 Service discovery introduction

Service discovery provides for the automated discovery of IP addresses used by remote iADT devices.

### 6.5.2 Service discovery using UDP

When an iADT port detects that the physical port inside the iADT port ~~transitions from unavailable to available~~has received an IP address it shall:

a) begin the ~~Service~~ service discovery broadcast processing (see 6.5.3);
b) continually receive packets on UDP port 4169; and
c) process service discovery messages received on UDP port 4169 (see 6.5.4).

The data octets of the user datagram shall contain the service discovery message defined in table x.

**Table x – Service discovery message**

| Bit<br>Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | SIGNATURE (“iADT”) | | | | | | | |
| 3 | | | | | | | | |
| 4 | INFORMATION TYPE | | | | | | | |
| 5 | DEVICE TYPE | | | | | | | |
| 6 | (MSB) | ADDITIONAL LENGTH | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | ADT MAJOR REVISION | | | ADT MINOR REVISION | | | | |
| 9 | UNSEC | TLS | Reserved | | | | | |
| 10 | Reserved | | | | | | | |
| 11 | Reserved | | | | | | | |

**~~Table x: Service discovery message~~**

| ~~Bit~~<br>~~Byte~~ | ~~7~~ | ~~6~~ | ~~5~~ | ~~4~~ | ~~3~~ | ~~2~~ | ~~1~~ | ~~0~~ |
|---|---|---|---|---|---|---|---|---|
| ~~0~~ | ~~INFORMATION TYPE~~ | | | | | | | |
| ~~1~~ | ~~DEVICE TYPE~~ | | | | | | | |
| ~~2~~ | ~~(MSB)~~ | ~~ADDITIONAL LENGTH~~ | | | | | | |
| ~~3~~ | | | | | | | | ~~(LSB)~~ |
| 4 | ~~ADT MAJOR REVISION~~ | | | ~~ADT MINOR REVISION~~ | | | | |
| | | | | | | | | |
| ~~10~~ | | | | | | | | |
| ~~11~~ | | | | | | | | |

The SIGNATURE field shall be set to the ASCII string “iADT”.

The INFORMATION TYPE field is defined in table y.

**Table y**

| Value | Description |
|---|---|
| 00h | Announcement |
| 01h | Response |
| 02h – FFh | Reserved |

The DEVICE TYPE field is defined in table z.

**Table z**

| Value | Description |
|---|---|
| 00h | DTD |
| 01h | Automation Device |
| 02h | Monitoring Application |
| 03h – FFh | Reserved |

The ADDITIONAL LENGTH field is the length of the data that follows. The value in the ADDITIONAL LENGTH field shall be a multiple of four.

The ADT MAJOR REVISION field set to the value that is used in the Port Login Information Unit (see 6.5.4).

The ADT MINOR REVISION field set to the value that is used in the Port Login Information Unit.

The UNSEC bit set to one indicates that the iADT port accepts iADT connections on port 4169.

The TLS bit set to one indicates that the iADT port accepts iADT connections on port <<TLS TBD>>.

### 6.5.3 Service discovery broadcast processing

The iADT device shall delay a random time in the range zero to three seconds and then broadcast the Service discovery message (see table x) once every three seconds until it receives a Service discovery message addressed to this ADT device or until a vendor specific number of messages have been sent. In the service discovery message~~:~~

a) the INFORMATION TYPE field shall be set to ~~announcement~~Anouncement (i.e., 00h~~;~~).
~~b)the DEVICE TYPE field shall be set:~~
 ~~A)to DTD if this device is a DTD device;~~
 ~~B)to automation device if this device is an automation device; or~~
 ~~C)to monitoring application if this is neither a DTD or automation device;~~
~~c)the ADDITIONAL LENGTH field shall be set to the length of the data that follows;~~
~~d)the ADT MAJOR REVISION field shall be set to the values that will be used in the Port Login Information Unit (see 6.5.4); and~~
~~e)the ADT MINOR REVISION field shall be set to the values that will be used in the Port Login Information Unit.~~

The purpose of the random delay is to avoid collisions between UDP packets from multiple devices.

### 6.5.4 Service discovery frame reception processing

When processing a service discovery message the ADT device shall check the fields to determine how to process the message.

If the INFORMATION TYPE field is set to announcement and the device type is not the same as this device, then the ADT device shall save the address information for the other device, delay a random time in the range zero to three seconds, and then send a service discovery message to that device in which:

a) the INFORMATION TYPE field shall be set to ~~response~~Response (i.e., 01h).~~;~~
~~b)the DEVICE TYPE field shall be set:~~
 ~~A)to DTD if this device is a DTD device;~~
 ~~B)to automation device if this device is an automation device; or~~
 ~~C)to monitoring application if this is neither a DTD or automation device;~~
~~c)the ADDITIONAL LENGTH field shall be set to the length of the data that follows;~~
~~d)the ADT MAJOR REVISION field shall be set to the value that is used in the Port Login Information Unit (see 6.5.4); and~~
~~e)the ADT MINOR REVISION field shall be set to the value that is used in the Port Login Information Unit.~~

If the INFORMATION TYPE field is set to ~~response~~ Response and the device type is not set to monitoring device and is not the same as this device, then stop the broadcast processing and save the address information for the other device.

~~Note: To prevent lots of devices from responding all at once, UPnP allows the control point (device looking for a service) to specify a range of time [0..120 sec.] to tell devices how long they have to respond. A device waits for a random time in that range before sending its response. We might incorporate that in the announcement message.~~

~~Note: Should we add a shutdown message?~~

~~Note: Should we specify a time to live (TTL) value?~~

### *Changes to clause 6*

Renumbered to clause 7 and changed as noted below.

The following contains new text in 7.2.6 which is not visible while change tracking is enabled.

#### ~~6~~7.2.6 AER Control information unit

An AER Control IU may be sent by an automation device port to a DT device port to enable or disable AER reporting. The payload of an AER Control IU shall contain a VHF Data structure, with the bits set to one for each field that the device shall report a change. The values in VHF Data structure shall be maintained separately for each I_T nexus. Multiple-bit fields shall have either all of the bits of the field set to one or all of the bits in the field set to zero. A DT device that receives a multi-bit field containing at least one bit set to zero shall treat that entire field as set to zero. A DT device shall consider reserved bits as not supported for AER notification.

DT devices that do not support AER shall send a NAK IU in response with the PR bit set to zero and a status code of UNSUPPORTED FRAME TYPE FOR SELECTED PROTOCOL (see table 14).

Except as noted in this sub clause, DT devices that support AER shall respond to the receipt of an AER Control IU by sending an AER Control IU back to the automation device with the same x_origin and exchange ID values. The payload of the IU shall contain a VHF Data IU data structure. Each field that has been enabled for AER notification for the I_T nexus and is supported by the device shall have all bits in the field set to one. Each field that has been either disabled for AER notification for the I_T nexus or is not supported for AER notification by the device shall be set to zero. The default setting for all AER events in a DT device shall be zero.

All AER control fields shall be set to zero by the DT device at the start of the port login process when the AOE bit is set to one.

DT devices ports shall only send an AER Control IU in response to receiving an AER Control IU from an automation device port.

## ~~6~~7.3 ADT frame header

<…>

The X_ORIGIN bit shall be set to zero if the ~~automation device originates the exchange~~ device originating the exchange is acting as an automation device within that session. The X_ORIGIN bit shall be set to one if the ~~DT device originates the exchange~~ device originating the exchange is acting as a DT device within that session. This bit shall remain constant for all frames associated with a given exchange.

~~Note: Is this actually unambiguous?~~
~~Note: Do we need to specify how monitoring applications set X_ORIGIN?~~

<…>

#### ~~6~~7.5.4 Port login information unit

<…>

The BAUD RATE field indicates the speed that the ~~port's~~ physical interface in an sADT port shall run after completion of negotiation. The BAUD RATE field contains the desired nominal Baud rate divided by 100. All ports shall default to operating at 9 600 Baud at power-up and following error conditions that require reestablishment of the operating parameters (see 4.6.2). If a port receives a Port Login IU containing a baud rate value less than 9 600 it shall respond with a NAK IU with a status code of NEGOTIATION ERROR (see table 14).

In an iADT port, the BAUD RATE field shall contain a value of 0000h.

<…>

### ~~6~~7.5.5 Port logout information unit

After sending a Port Logout IU and before receiving the corresponding acknowledgement IU, a port may discard without acknowledgement any frame, other than an acknowledgement IU, received.

Upon receiving a Port Logout IU, a DT Device port shall:

a) abort all open exchanges;
b) disable Asynchronous Event Reporting;
c) disable initiating Port Login exchanges; ~~and~~
d) set port operating parameters to default following transmission of the ACK IU for the Port Logout IU (see 4.2) ; and
d) set the logout duration time to the value in the LOGOUT DURATION field. If the value in the field is zero, then set the logout duration value to infinite.

Upon receiving a Port Logout IU, an automation port shall:

a) abort all open exchanges;
b) disable initiating Port Login exchanges; ~~and~~
c) set port operating parameters to default following transmission of the ACK IU for the Port Logout IU (see 4.2); and
d) set the logout duration time to the value in the LOGOUT DURATION field. If the value in the field is zero, then set the logout duration value to infinite.

If a DTD port sends a Port Logout IU to an automation port, then it should send a Port Login IU to the automation port within the logout duration time.

Knowledge of the logged out state may be volatile, as a result of a hard reset condition in the logged out port may cause the port to become active again and attempt to log in to the attached port.

The payload of the Port Logout IU is shown in table 16.

**Table 16: Port Logout IU payload contents**

| Bit<br>Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | LOGOUT DURATION | | | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | ESR | REASON CODE | | | | | | |
| 3 | Reserved | | | | | | | |

The LOGOUT DURATION field contains the value to which the port that receives the Port Logout IU shall set its logout duration time, i.e., the length in seconds that ~~the port that receives the Port Logout IU~~ it shall remain in P3:Logged-out state. A value of zero indicates that the port that receives the Port Logout IU shall set its logout duration time to infinite, i.e., it shall remain in the P3:Logged-out state until it receives a Port Login IU.

The external stimulus required (ESR) bit set to one indicates that the port requires an external stimulus before initiating a negotiation exchange (see 6.5.13.1).

<…>

Editor's note: While the paragraph below was not in the previous revision of the proposal, I am showing it in black so that the proposed changes will stand out.

After a port sends an ACK IU in response to a Port Logout IU it shall set its operating parameters to default ~~and~~, enter the P3:Logged-Out state, and if it is an iADT port, close the connection. Once the originator of a Port Logout IU receives an ACK IU for that exchange, it shall set its operating parameters to default ~~and~~, enter the P0:Initial state, and if it is an iADT port, close the connection. See 4.3 for a definition of the port states.

Note: Do we need an explicit statement here that the session is ended? That is already stated several paragraphs further down.

~~Note 1: Does the session concept need to be explained here?~~
~~Note 2: Should we recommend against closing the connection to a drive performing bridging?~~

## 7.6 Sessions

A session is a relationship between two ADT ports. The link layer manages sessions and associates connections with sessions. A session between two sADT ports is identified by two physically-connected sADT ports. A session between two iADT ports is identified by a unique pair of IP addresses.

A session between a local sADT port and a remote sADT port begins in the local sADT port when no session exists between the two sADT ports and the local sADT port transmits or receives a Login IU.

A session between a local iADT port and a remote iADT port begins in the local iADT port when no session exists between the two iADT ports and:

a) the link layer invokes the **Connect** service request specifying the remote iADT port; or
b) the connection layer invokes the **Connect Received** service indication specifying the remote iADT port.

A session between two ADT ports ends in the local ADT port after the local ADT port processes a Logout IU (see 7.5.5) or an implicit logout message (see 4.3.2.4.4).

In an iADT port, if:

1) a session exists;
2) a Login IU has been received; and
3) an new connection is established in the same session

then the previously-established connection shall be closed.

In an iADT port, if:

1) a session exists;
2) a Login IU has not been received; and
3) an new connection is established in the same session

then the connection initiated by the DTD shall be closed.

Editor's Note: The above lists must indicate sequence.

Annex E contains examples of resolving redundant connections.

*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~~exchanged between an automation device transport layer and a DTD transport layer (see Figure 5). 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.

Editor's Note: We have agreed that fast access data will be maintained on a per-I_T nexus basis.

<…>

*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;<br>b) L is used to set the LUN field; and<br>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:<br>a) I_T is indicated by the session and the X_ORIGIN field;<br>b) L is used to set the LUN field; and<br>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:<br>a) I_T is indicated by the session and the X_ORIGIN field;<br>b) L is used to set the LUN field; and<br>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. |
| <…> | |

<…>

***New Annex D***

# Annex D
# (informative)

## iADT Connection Services Relationship to Sockets API

In TCP/IP implementations, the TCP calls mentioned in clause 6 are typically invoked via a Sockets application programming interface (API). The details of the Sockets API varies between implementations. This annex describes the typical semantics of the Sockets API function calls and how the connection services may be mapped to those function calls.

Table D.1 describes the function calls in a typical Sockets API.

**Table D.1 – 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. Performs a TCP active OPEN and causes entry to the SYN-SENT state. Return from connect() indicates transition to the ESTABLISHED state. |
| **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. listen() performs a TCP passive OPEN and causes entry to the LISTEN state.<br>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.<br>The original socket may be used to accept additional connection requests.<br>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.<br>An error status on return from send() may indicate entry to the CLOSE-WAIT or FIN-WAIT-1 state. |
| **recv()** | Receives incoming data that has been received by a connected socket.<br>An error status on return from recv() may indicate entry to the CLOSE-WAIT or FIN-WAIT-1 state. |
| **shutdown()** | Closes a connection, optionally preventing further sends and/or receives. Causes a transition from the ESTABLISHED state to the FIN-WAIT-1 state and eventually causes a transition to the CLOSED state. |
| **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.<br>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 D.2 shows how connection services may be mapped to Sockets API function calls. The **Reset** service request and **~~Reset received~~Reset Received** service indication are not listed because they are not relevant to the Sockets API.

**Table D.2 – Connection service mapping to Sockets API functions**

| Connection service | Socket function | Notes |
|---|---|---|
| **Connect** | socket() | |
| | bind() | As part of processing a **connect** service request, 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. |
| **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 is invoked after the send() returns. |
| **Receive** | recv() | Invocation of the **Receive** service request causes the invocation of recv(), which blocks 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() | The **Disconnect** service request causes invocation of the shutdown() |
| | close() | close() deallocates the socket used for the TCP connection. |
| **Disconnected** | shutdown() return | Completion of closing of a TCP connection which was initiated by the local port causes a return from close(). This causes invocation of the **Disconnected** service confirmation.<br>Other events (e.g., physical port failure) may also cause invocation of **Disconnected**. |
| **~~Disconnect received~~Disconnect Received** | send() return<br>recv() return | Closing of a TCP connection by the remote port causes invocation of the **~~Disconnect received~~Disconnect Received** service indication. |
| | close() | close() deallocates the socket used for the TCP connection. |

*New Annex E*

# Annex E
# (informative)

# iADT Redundant Connection Resolution Examples

## E.1 Introduction

If two iADT ports initiate TCP connections simultaneously, then one of the connections is closed as specified in ~~Table 14a~~7.6. The order in which each connection is established may appear differently on each iADT port depending upon transmission delays. This annex provides examples to illustrate the application of the rules in ~~Table 14a~~7.6.

~~In the figures in this annex, the labels “AD” and “DTD” indicate that the connection was initiated by the automation device or the DTD, respectively.~~Figure E.1 shows the drawing conventions used in this annex. The communications associated with each connection are distinguished by use of color and solid or dashed lines. A vertical line with two horizontal lines adjacent to the Link Layer symbol indicates the beginning and end of a session. A shaded box connecting the link layers in each device and labeled “Login Processing” indicates when login processing is performed.

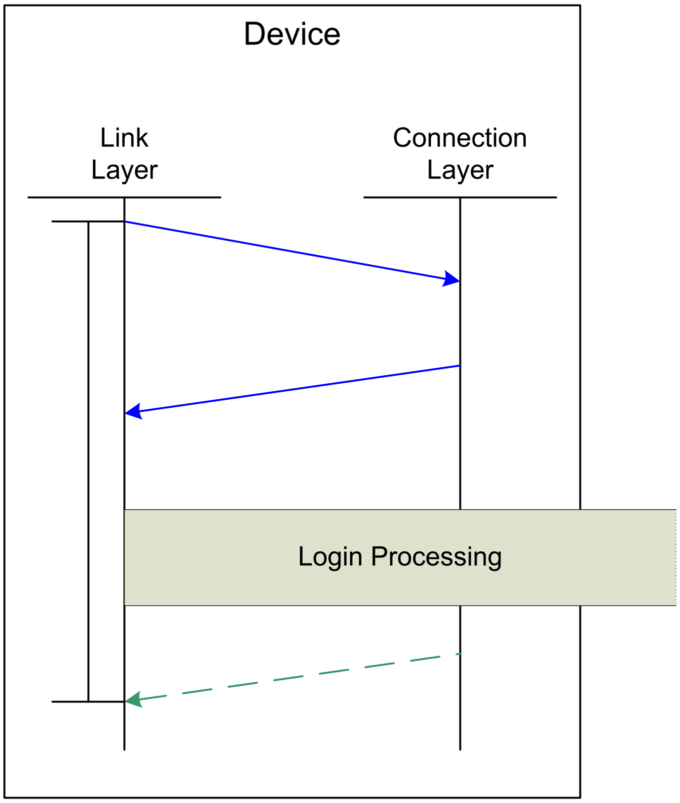


**Figure E.1 – Examples of session lifetimes**

## E.2 Ambiguous connection establishment order

Figure E.~~1~~ 2 shows the case in which each device sees the connections established in a different order~~,~~ (i.e., in the automation device the connection initiated by the DTD is established first, and in the DTD the connection initiated by the automation device is established first). No login is processed during the actions shown in this figure. Following the rules in ~~Table 14a~~7.6, the automation device ~~takes no action~~ and the DTD ~~closes the connection it initiated~~close the connection initiated by the DTD.

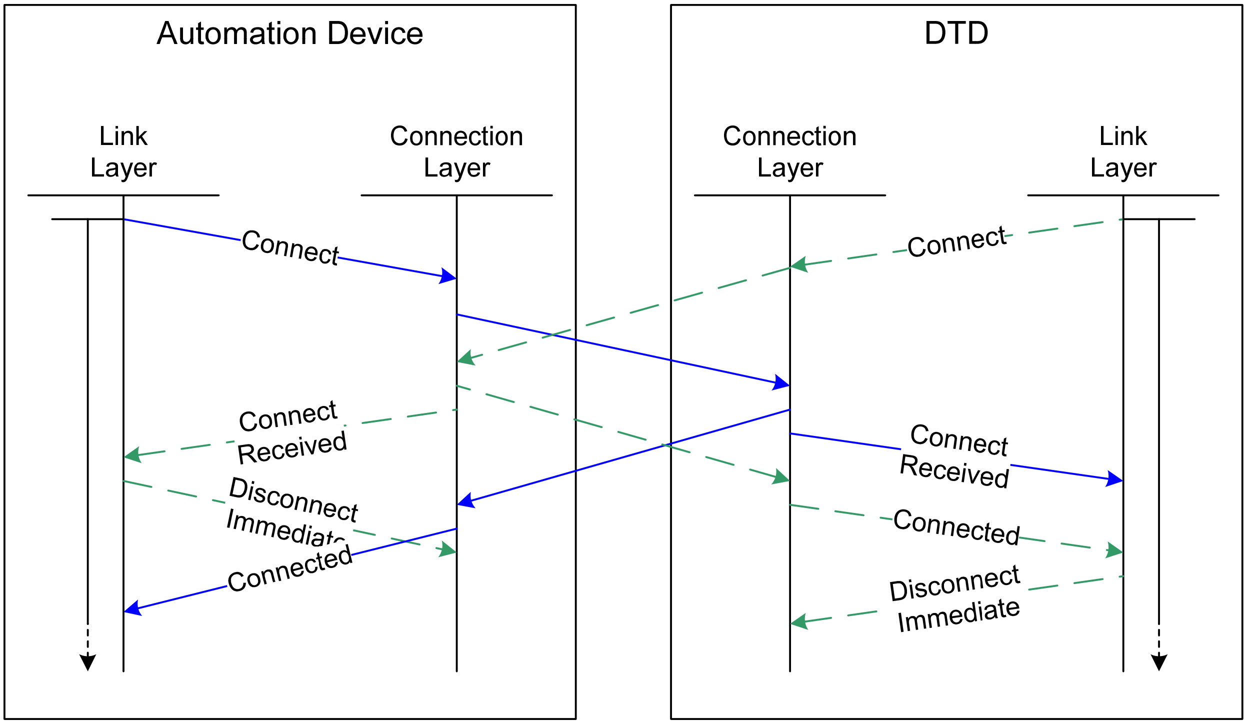


**Figure E.~~1~~ 2 - Ambiguous connection establishment order**

## E.3 Automation device-initiated connection established first

Figure E.~~2~~ 3 shows the case in which both devices see the connection initiated by the automation device being established first. No login is processed during the actions shown in this figure. Following the rules in ~~Table 14a~~7.6, the automation device ~~takes no action~~ and the DTD ~~closes the connection it initiated~~close the connection initiated by the DTD.

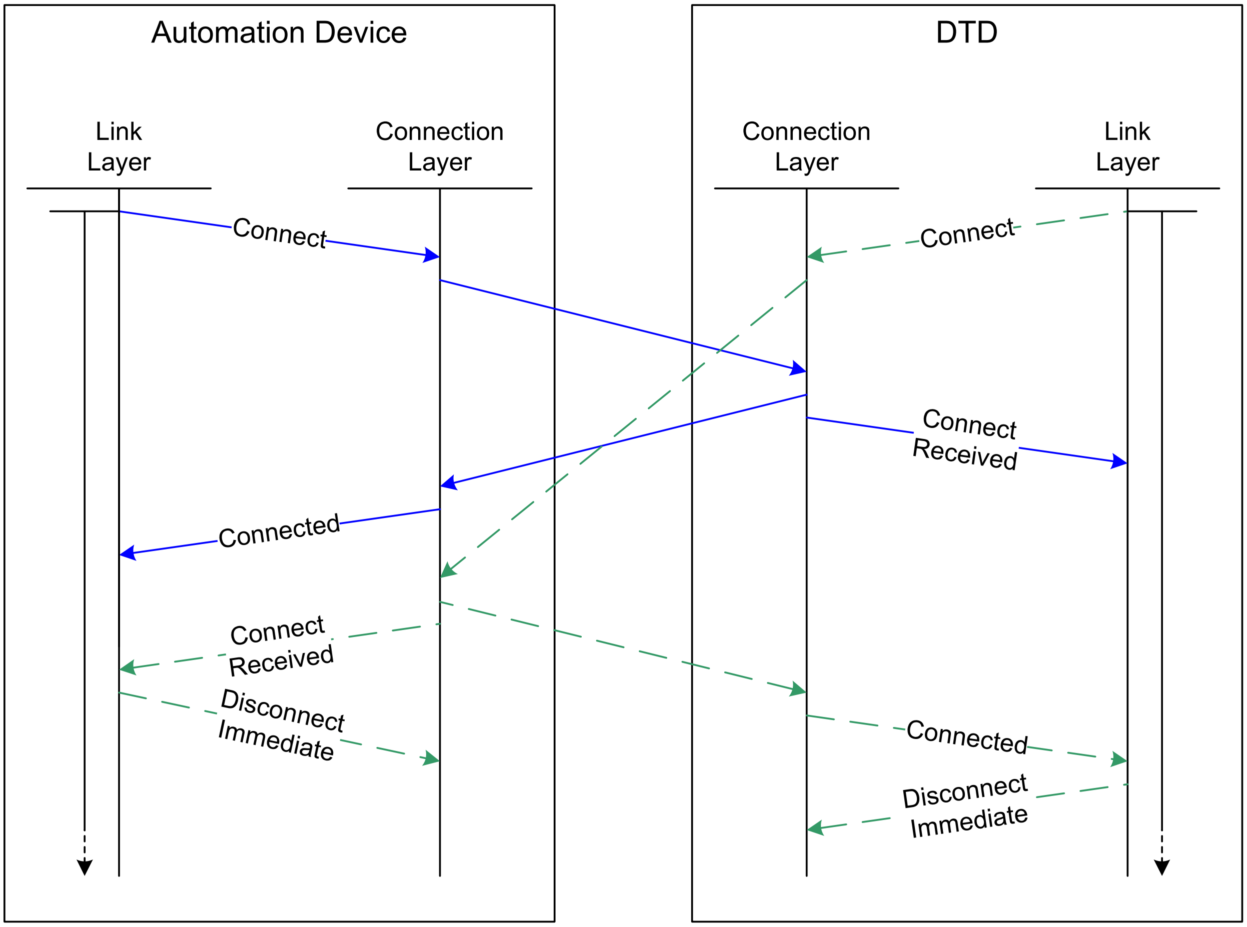


Figure E.~~2~~ 3 - Automation device-initiated connection established first

## E.4 DTD-initiated connection established first

Figure E.~~3~~ 4 shows the case in which both devices see the connection initiated by the DTD being established first. No login is processed during the actions shown in this figure. Following the rules in ~~Table 14a~~7.6, the automation ~~device takes no action~~and the DTD ~~closes the connection it initiated~~close the connection initiated by the DTD.

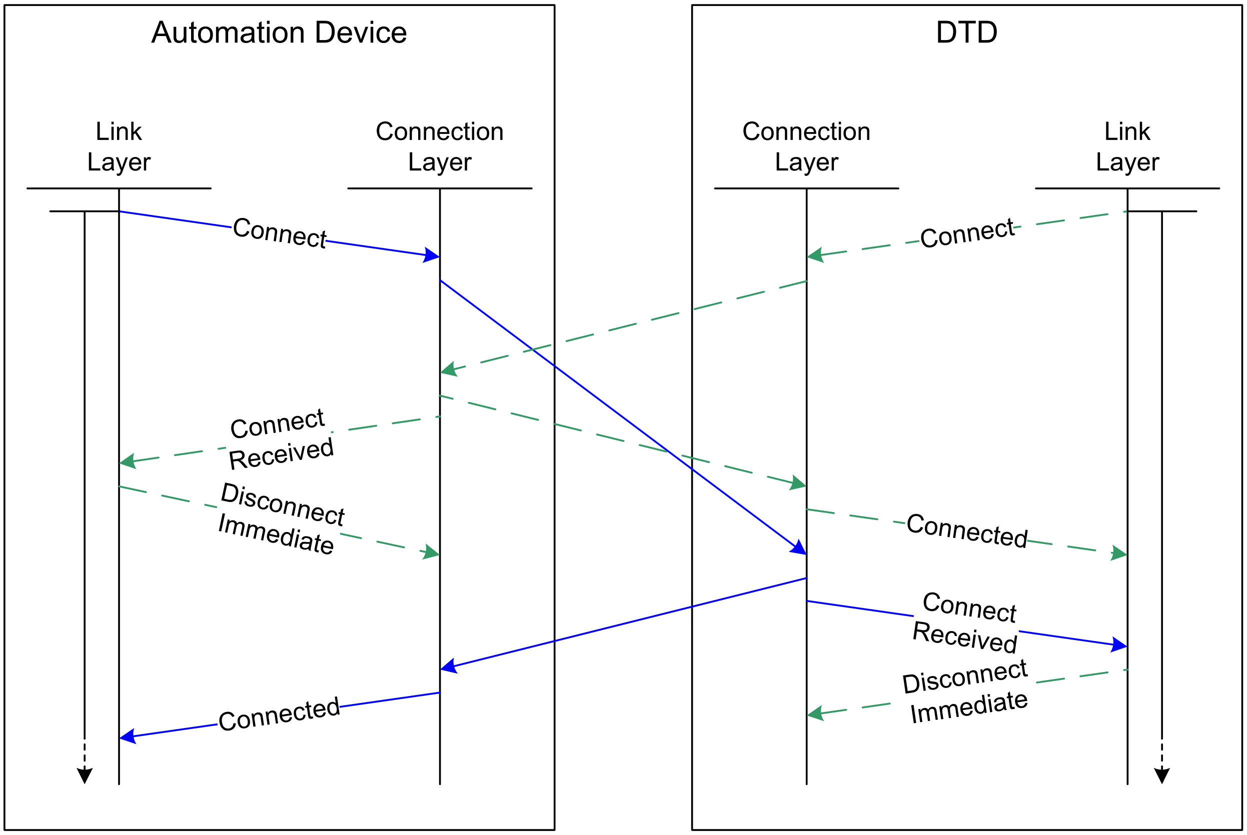


**Figure E.~~3~~ 4 - DTD-initiated connection established first**

## E.5 Duplicate automation device-initiated connections

Figure E.~~4~~ 5 shows the case in which ~~the automation~~one device initiates ~~two connections in succession~~a connection, login is processed, and the same device initiates a second connection without closing the first connection. ~~This behavior is prohibited, but this example illustrates how the DTD should respond.~~ Following the rules in ~~Table 14a~~7.6, ~~the automation device takes no action and the DTD closes~~ the first connection established is closed.

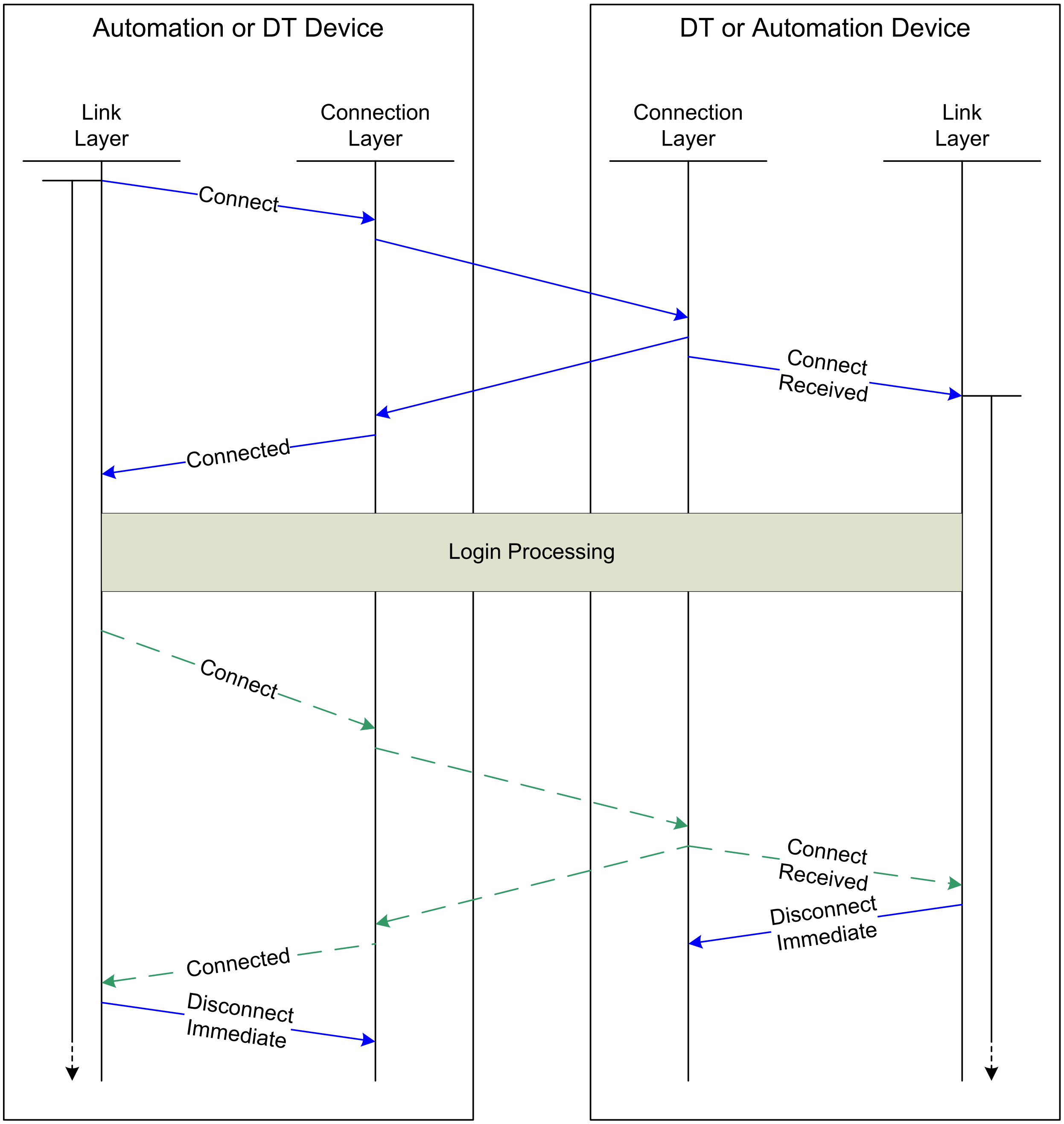


**Figure E.~~4~~ 5 – Duplicate automation device-initiated connections**

## E.6 Duplicate DTD-initiated connections

Figure E.~~5~~ 6 shows the case in which ~~the DTD initiates two connections in succession~~one device initiates a connection, login is processed, and the other device initiates a connection without closing the first connection. ~~This behavior is prohibited, but this example illustrates how the automation device should respond.~~ Following the rules in ~~Table 14a~~7.6, ~~the DTD takes no action and the automation device closes~~ the first connection established is closed.

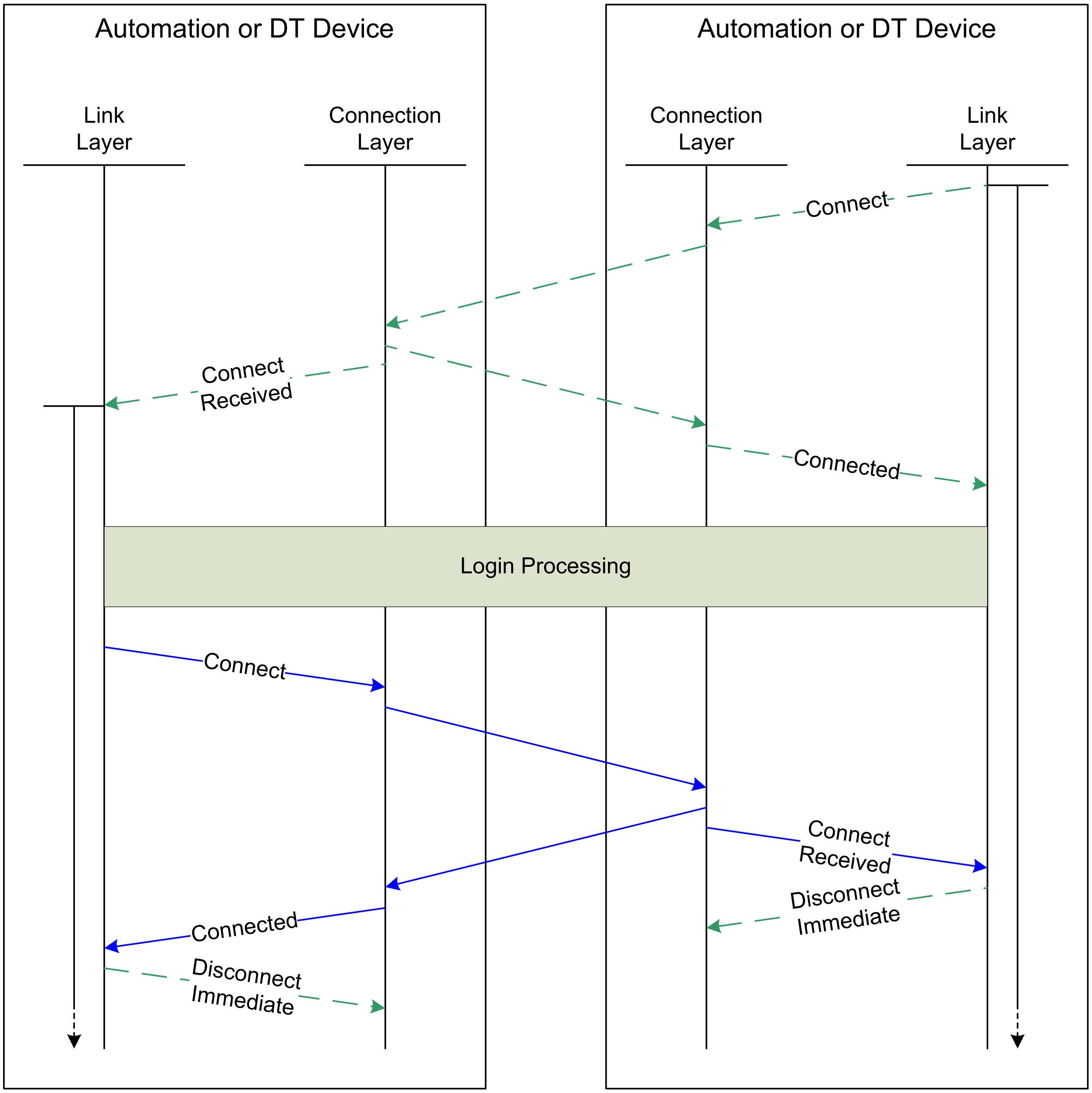


Figure E.~~5~~ 6 – Duplicate DTD-initiated connections