Working Draft American National Standard

Project T13/1700-D

Revision 0
22 February 2005

Information technology - AT Attachment with Packet Interface - 8 Architecture Model (ATA/ATAPI-8 AM)

T13 Technical Editor:

Mark Evans
Maxtor Corporation
500 McCarthy Boulevard
Milpitas, CA 95035
USA

Telephone: 408-894-5310
Email: mark_evans@maxtor.com

Reference number
ISO/IEC ***:200x**
ANSI INCITS.*:200x**

Points of Contact

International Committee for Information Technology Standards (INCITS) T13 Technical Committee

T13 Chair
Daniel Colegrove
Hitachi GST
2903 Carmelo Drive
Henderson, NV 89502
USA

Telephone: 702-614-6119
Email: dc

T13 Vice-chair
James Hatfield
Seagate Technology
389 Disc Drive
Longmont, CO 80503
USA

Telephone: 202-737-8888
Email: jh

T13 Web Site:

http://www.t13.org

T13 Reflector:

See the T13 web site for reflector information.

INCITS Secretariat:
INCITS Sectretariat
Suite 200
1250 Eye Street, NW
Washington, DC 20005
USA

Telephone: 202-737-8888
Web site: http://www.incits.org
Email: incits@itic.org

Document Distribution:
INCITS Online Store
managed by Techstreet
1327 Jones Drive
Ann Arbor, MI 48105
USA

Web site: http://www.techstreet.com/incits.html
Telephone: 734-302-7801 or 800-699-9277

Global Engineering Documents, an IHS Company
15 Inverness Way East
Englewood, CO 80112-5704
USA

Web site: http://global.ihs.com
Telephone: 303-397-7956 or 303-792-2181 or 800-854-7179

American National Standard
for Information Technology

AT Attachment with Packet Interface - 8
Architecture Model

Secretariat

Information Technology Industry Council

Approved: mm/dd/yyyy

American National Standards Institute, Inc.

Abstract

This standard specifies the AT Attachment Architectural Model. The purpose of the architecture model is to provide a common basis for the coordination of ATA standards and to specify those aspects of ATA system behavior that are independent of a particular technology and common to all implementations.

Published by:

American National Standards Institute
11 W. 42nd Street, New York, New York 10036

Printed in the United States of America

Revision History

R.1 Revision AA8-AMr00 (22 February 2005)

First release of AA8-AM.

Contents

Page

Tables

Figures

Page

Foreword

This foreword is not part of American National Standard INCITS.***:200x.

The purpose of this standard is to provide a basis for the coordination of ATA standards development and to define requirements, common to all ATA technologies and implementations, that are essential for compatibility with host ATA application software and device-resident firmware across all ATA transport protocols. These requirements are defined through a reference model that specifies the behavior and abstract structure that is generic to all ATAI I/O system implementations.

With any technical document there may arise questions of interpretation as new products are implemented. INCITS has established procedures to issue technical opinions concerning the standards developed by INCITS. These procedures may result in ATA Technical Information Bulletins being published by INCITS.

These Bulletins, while reflecting the opinion of the Technical Committee that developed the standard, are intended solely as supplementary information to other users of the standard. This standard, ANSI INCITS.***:200x, as approved through the publication and voting procedures of the American National Standards Institute, is not altered by these bulletins. Any subsequent revision to this standard may or may not reflect the contents of these Technical Information Bulletins.

Current INCITS practice is to make Technical Information Bulletins available through:

INCITS Online Store
managed by Techstreet
1327 Jones Drive
Ann Arbor, MI 48105 USA

Web site: http://www.techstreet.com/incits.html
Telephone: 734-302-7801 or 800-699-9277
Facsimile: 734-302-7811

or

Global Engineering Documents, an IHS Company
15 Inverness Way East
Englewood, CO 80112-5704 USA

Web site: http://global.ihs.com
Telephone: 303-397-7956 or 303-792-2181 or 800-854-7179
Facsimile: 303-792-2192

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, National Committee for Information Technology Standards, Information Technology Institute, 1250 Eye Street, NW, Suite 200, Washington, DC 20005- 3922.

This standard was processed and approved for submittal to ANSI by the InterNational Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time of it approved this standard, INCITS had the following members:

<<Insert INCITS member list>>

The INCITS Technical Committee T13 on ATA Storage Interfaces, that reviewed this standard, had the following members:

<<Insert T13 member list>>

Introduction

The AT Attachment-8 Architecture Model (ATA8-AM) standard is divided into the following clauses and annexes:

Clause 1 is the scope.
Clause 2 enumerates the normative references that apply to this standard.
Clause 3 describes the definitions, symbols, and abbreviations used in this standard.
Clause 4 describes the overall ATA architectural model.
Clause 5 describes the ATA command model element of the ATA architecture.
Clause 6 describes the events that may be detected by a ATA device.

American National Standard -
For Information Technology

AT Attachment with Packet Interface - 8 Architecture Model (ATA/ATAPI-8 AM)

1 Scope

1.1 Introduction

The set of AT Attachment standards consists of this standard and the ATA implementation standards described in 1.3. This standard defines a reference model that specifies common behaviors for ATA hosts and devices and an abstract structure that is generic to all ATA I/O system implementations.

The set of ATA standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming ATA implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

1.2 Requirements precedence

This standard defines generic requirements that pertain to ATA implementation standards and implementation requirements. An implementation requirement specifies behavior in terms of measurable or observable parameters that apply to an implementation. Examples of implementation requirements defined in this document are [editor's note: examples from this standard are needed here].

Generic requirements are transformed to implementation requirements by an implementation standard. An example of a generic requirement is [editor's note: an example from this standard is needed here].

Figure 1 shows the precedence of requirements for the ATA standards relative to ATA implementations.

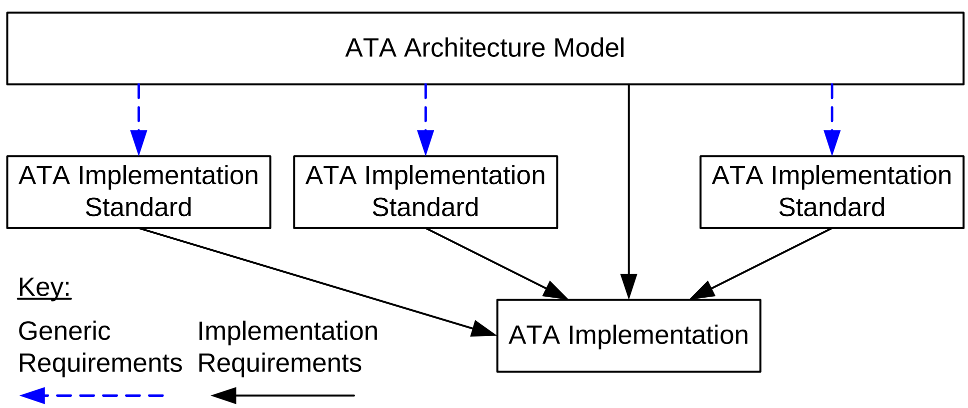

Figure 1 — Requirements precedence

As shown in figure 1, all ATA implementation standards shall reflect the generic requirements defined in this standard. In addition, an implementation claiming ATA compliance shall conform to the applicable implementation requirements defined in this standard and the appropriate ATA implementation standards. In the event of a conflict between this document and other ATA standards under the jurisdiction of technical committee T13, the requirements of this standard shall apply.

1.3 ATA family of standards

Figure 2 shows the relationship of this standard to the other standards and related projects in the ATA and SCSI families of standards and specifications.

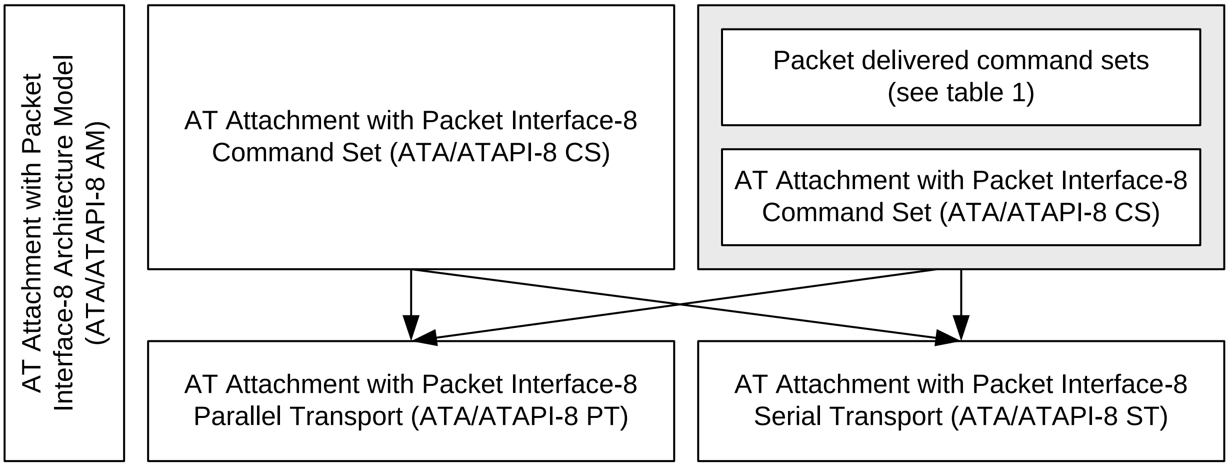

Figure 2 — Standards relationships

Figure 2 is not intended to imply a relationship such as a hierarchy, protocol stack, or system architecture.

The functional areas identified in figure 2 characterize the scope of standards within a group as follows:

AT Attachment with Packet Interface-8 Architecture Model: defines the ATA systems model, the functional partitioning of the ATA and SCSI standard sets relative to ATA implementation, and requirements applicable to all ATA implementations and implementation standards.

AT Attachment with Packet Interface-8 Command Set: defines the command set for devices implementing the register delivered command set and for devices implementing the PACKET command feature set (commonly known as ATAPI devices).

Packet delivered command sets: defines the command sets defining commands that may be delivered using the PACKET command feature set. Table 1 shows packet delivered command sets.

Table 1 — Packet delivered command sets

SCSI Primary Commands (SPC)
SCSI Primary Commands - 2 (SPC-2)
SCSI Primary Commands - 3 (SPC-3)
SCSI Block Commands (SBC-2)
SCSI Stream Commands (SSC)
Multimedia Commands (MMC)
Multimedia Commands - 2 (MMC-2)
Multimedia Commands - 3 (MMC-3)
Multimedia Commands - 4 (MMC-4)
ATAPI for Removable Media (SFF-8070I)
ATA Packet Interface (ATAPI) for Streaming Tape QIC-157 revision D

AT Attachment with Packet Interface-8 Parallel Transport: defines the following items for the parallel ATA interface:

a) The connectors and cables for physical interconnection between host and storage device;
b) The electrical characteristics of the interconnecting signals;
c) The logical characteristics of the interconnecting signals
d) The protocols for transporting commands, data, and status using the interface.

AT Attachment with Packet Interface-8 Serial Transport: defines the following items for the serial ATA interface:

a) The connectors and cables for physical interconnection between host and storage device;
b) The electrical characteristics of the interconnecting signals;
c) The logical characteristics of the interconnecting signals
d) The protocols for transporting commands, data, and status using the interface.

2 Normative references

2.1 Normative references overview

The following standards contain provisions that, by reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the following documents may be obtained from ANSI:

a) approved ANSI standards;
b) approved and draft international and regional standards (ISO, IEC, CEN/CENELEC, ITUT); and
c) approved and draft foreign standards (including BSI, JIS, and DIN).

For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.

Additional availability contact information is provided below as needed. Table 2 shows standards and specifications bodies and their web sites.

Table 2 — Standards bodies

Abbreviation	Standards or specifications body	Web site
ANSI	American National Standards Institute	http://www.ansi.org
BSI	British Standards Institution	http://www.bsi-global.com
CEN	European Committee for Standardization	http://www.cenorm.be
CENELEC	European Committee for Electrotechnical Standardization	http://www.cenelec.org
DIN	German Institute for Standardization	http://www.din.de
IEC	International Engineering Consortium	http://www.iec.ch
IEEE	Institute of Electrical and Electronics Engineers	http://www.ieee.org
INCITS	International Committee for Information Technology Standards	http://www.incits.org
ISO	International Standards Organization	http://www.iso.ch
ITI	Information Technology Industry Council	http://www.itic.org
ITU-T	International Telecommunications Union Telecommunications Standardization Sector	http://www.itu.int
JIS	Japanese Industrial Standards Committee	http://www.jisc.org
T10	INCITS T10 SCSI storage interfaces	http://www.t10.org
T11	INCITS T11 Fibre Channel interfaces	http://www.t11.org
T13	INCITS T13 ATA storage interface	http://www.t13.org
SFF	SFF Committee	http://sffcommittee.org

2.2 Approved references

At the time of publication, the following referenced standards or technical reports had been approved:

ISO/IEC 13213:1994 *Control and Status Register (CSR) Architecture for microprocessor buses*
ISO/IEC 14776-452, *SCSI Primary Commands - 2 (SPC-2)* [ANSI NCITS.351-2001]

SCSI-3 *Block Commands (SBC)* [ANSI INCITS 306-1998]
SCSI-3 Streaming Commands (SSC) [ANSI INCITS 335-2000]

Multimedia Commands (MMC) [ANSI X3.304-1997]
Multimedia Commands - 2 (MMC-2) [ANSI INCITS 333-2000]
Multimedia Commands - 3 (MMC-3) [ANSI INCITS 360-2002]
Protected Area Run Time Interface Extensions (PARTIES) [ANSI INCITS 346-2001]
SCSI Primary Commands - 2 (SPC-2) [ANSI INCITS 351-2001]
AT Attachment with Packet Interface Extension (ATA/ATAPI-4) [ANSI INCITS.317-1998]

[editor's note: we need to decide on which standards and specifications should be referenced, and then get the correct numbers for them.]

2.3 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the document, or regarding availability, contact the relevant standards body or other organization as indicated.

ISO/IEC 14776-352, SCSI Media Changer Commands - 2 (SMC-2) standard (T10/1383-D)
ISO/IEC 14776-364, SCSI Multimedia Commands - 4 (MMC-4) standard (T10/1545-D)
ISO/IEC 14776-372, SCSI Enclosure Services - 2 (SES-2) standard (T10/1559-D)
ISO/IEC 14776-413, SCSI Architecture Model - 3 (SAM-3) standard (T10/1561-D)
ISO/IEC 14776-453, SCSI Primary Commands - 3 (SPC-3) standard (T10/1416-D)

2.4 Other references

The following standards and specifications are also referenced.

PC Card Standard, February 1995, PCMCIA (68-pin Connector)

For the PC Card Standard published by the Personal Computer Memory Card International Association, contact PCMCIA at 408-433-2273 or http://www.pc-card.org.

CompactFlash™ Association Specification, Revision 1.4

For the CompactFlash™ Association Specification published by the CompactFlash? Association, contact the CompactFlash™ Association at http://www.compactflash.org.

ATA Packet Interface (ATAPI) for Streaming Tape QIC-157 revision D

For QIC specifications published by Quarter-Inch Cartridge Drive Standards, Inc., contact them at 805 963-3853 or http://www.qic.org.

For more information on the current status of SFF document, contact the SFF Committee at 408-867-6630 (phone), or 408-867-2115 (fax). To obtain copies of these documents, contact the SFF Committee at 14426 Black Walnut Court, Saratoga, CA 95070 at 408-867-6630 (phone) or 408-741-1600 (fax) or see http://www.sffcommittee.org.

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1 application client: an object in a host that is the source of application client tasks (see 4.2.2).

3.1.2 application client task: an object created by an application client to process a single command or device management operation (see 4.2.2).

3.1.3 command: a unit of work performed by a device that is not a device management function (e.g., a data transfer operation).

3.1.4 device: a storage peripheral that processes commands and device management functions (see 4.2.3).

3.1.5 device management function: a function performed by a device that is not a command (e.g., a reset function).

3.1.6 device manager: an object in a device that controls the sequencing of one or more commands and processes device management functions (see 4.2.3).

3.1.7 device port: an object in a device that acts as the connection between its device server and device manager and the service delivery subsystem (see 4.2.3).

3.1.8 device server: an object in a device that processes commands (see 4.2.3).

3.1.9 domain: an I/O subsystem that is made up of one host, one or more devices, and a service delivery subsystem (see 4.2.1).

3.1.10 host: an object that originates commands and device management functions (see 4.2.2).

3.1.11 Host port: an object in a host that acts as the connection between its application client(s) and the service delivery subsystem (see 4.2.2).

3.1.12 logical block: a set of data bytes accessed and referenced as a unit.

3.1.13 logical block address (LBA): the value used to reference a logical block.

3.1.14 read data: data transferred from a device port to a host port in response to a command.

3.1.15 write data: data transferred from a host port to a device port in response to a command.

3.2 Symbols

3.3 Abbreviations

3.4 Keywords

3.4.1 expected: a keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

3.4.2 ignored: a keyword used to describe an unused bit, byte, word, field, or code value. The contents or value of an ignored bit, byte, word, field, or code value may be set to any value by the sender and shall not be examined by the recipient.

3.4.3 invalid: a keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field, or code value shall be reported as an error.

3.4.4 mandatory: a keyword used to describe an item that is required to be implemented as defined in this standard.

3.4.5 may: a keyword that indicates flexibility of choice with no implied preference (synonymous with “may or may not”).

3.4.6 may or may not: a keyword that indicates flexibility of choice with no implied preference (synonymous with “may”).

3.4.7 obsolete: a keyword used to describe bits, bytes, words, fields, and code values that may have been defined in previous standards are not defined in this standard and shall not be reclaimed for other uses in future standards. However, some degree of functionality may be required for items designated as “obsolete” to provide for backward compatibility.

3.4.8 optional: a keyword used to describe a feature that is not required to be implemented by this standard. However, if any optional feature defined in this standard is implemented, then it shall be implemented as defined in this standard.

3.4.9 prohibited: a keyword used to describe a feature that shall not be implemented.

3.4.10 reserved: a keyword used to describe bits, bytes, words, fields, and code values that are set aside for future standardization. The use and interpretation of reserved bits, bytes, words, fields, and code values may be specified by future extensions to this or other standards. A reserved bit, byte, word, or field shall be cleared to zero, or in accordance with a future extension to this standard. The recipient shall not check reserved bits, bytes, words, or fields. Receipt of reserved code values in defined fields shall be treated as a command parameter error and reported by returning command aborted.

3.4.11 restricted: a keyword used to describe bits, bytes, words, fields, and code values that are set aside for use in other standards or for other data structures in this standard. A restricted bit, byte, word, field, or code value shall be treated as a reserved bit, byte, word, field, or code value for the purposes of the requirements defined in this standard.

3.4.12 retired: a keyword used to describe bits, bytes, words, fields, and code values that had been defined in previous standards but are not defined in this standard. Retired bits, bytes, words, fields, and code values may be reclaimed for other uses in future standards. If retired bits, bytes, words, fields, or code values are used before they are reclaimed, they shall have the meaning or functionality as described in previous standards.

3.4.13 shall: a keyword used to describe a mandatory requirement. (“Shall” is synonymous “is required to”.) Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.4.14 should: a keyword indicating flexibility of choice with a strongly preferred alternative. (“Should” is synonymous with the phrase “it is strongly recommended”.)

3.4.15 vendor specific: a keyword used to describe an item (e.g., a bit, field, or code value) that is not defined by this standard and may be used differently in various implementations.

3.5 Conventions

3.5.1 Editorial conventions

Lowercase is used for words having the normal English meaning. Certain words and terms used in this 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.

The names of abbreviations, commands, fields, and acronyms used as signal names are in all uppercase (e.g., IDENTIFY DEVICE). Fields containing only one bit are usually referred to as the “name” bit instead of the “name” field.

Wherever possible in this standard the term “specify” or any of its forms denotes an action by the host (e.g., a host specifies a command code). Wherever possible in this standard the term “indicate” or any of its forms denotes an action by the device (e.g., a device indicates status).

[editor’s note: try to add some more words to clarify this]

Wherever possible in this standard the term “send” or any of its forms denotes an action internal to a host or device (e.g., an application client sends a protocol service request to a host port). Wherever possible in this standard the term “transmit” or any of its forms denotes an action external to a host or device (e.g., a host port transmits data to a device port).

[editor’s note: I looked for an analog for “receive” and didn’t find a good word.]

3.5.2 Numeric conventions

A binary number is represented in this standard by any sequence of digits comprised of only the Western-Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores or spaces may be included between characters in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits comprised of only the Western-Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included between characters in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h or B_FD8C_FA23h).

A decimal number is represented in this standard by any sequence of digits comprised of only the Western-Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

This standard uses the American convention for representing decimal numbers (e.g., the thousands and higher multiples are separated by a comma, and a decimal point is used to separate the ones and higher digits and the tenths and smaller digits). Table 3 shows some examples of decimal numbers using the American and ISO numbering conventions.

Table 3 — American and ISO numbering conventions

American	ISO
0.6	0,6
3.14159265	3.141 592 65
1,000	1 000
1,323,462.95	1 323 462,95

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., $666.\overline{6}$ means 666.666666... or 666 2/3, and $12.\overline{142857}$ means 12.142857142857... or 12 1/7).

3.5.3 List notation

Ordered lists showing an ordering relationship between the listed items are shown in this standard in the form of:

1) First occurrence;
2) Second occurrence; and
3) Third occurrence.

Unordered lists showing no ordering relationship between the listed items are shown in this standard in the form of:

a) Item a;
b) Item b; or (or and)

c) Item c.

Unordered lists may be nested within ordered or unordered lists. A nested unordered list within an ordered list is shown in this standard in the form of:

a) Item a:
   A) Item A;
   B) Item B; or (or and)
   C) Item C;
b) Item b; or (or and)
c) Item c.

3.5.4 Precedence

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

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

4 ATA architecture model

4.1 Introduction

The purpose of the ATA architecture model is to:

a) Provide a basis for the coordination of ATA standards development that allows each standard to be placed into perspective within the overall ATA architecture model;
b) Establish a layered model by which standards may be developed;
c) Provide a common reference for maintaining consistency among related standards; and
d) Provide the foundation for application compatibility across all ATA interconnect and transport protocol environments by specifying generic requirements that apply uniformly to all implementation standards within each functional area.

The development of this standard is assisted by the use of an abstract model. To specify the external behavior of an ATA system, elements in a system are replaced by functionally equivalent components within this model. Only externally observable behavior is retained as the standard of behavior. The description of internal behavior in this standard is provided only to support the definition of the observable aspects of the model. Those aspects are limited to the generic properties and characteristics needed for host applications to interoperate with hosts and devices in any ATA interconnect and transport protocol environment. The model does not address other requirements that may be essential to some I/O system implementations (e.g., the mapping of device addresses, the procedure for discovering devices on a network, and the definition of network authentication policies). These considerations are outside the scope of this standard.

The set of ATA standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming ATA implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

The ATA architecture model is described in terms of objects, protocol layers, and service interfaces between objects. As used in this standard, objects are abstractions, encapsulating a set of related functions, data types, and other objects.

Certain objects are defined by ATA. For example, the Command object as described in this standard is instantiated in the parallel ATA transport protocol by the writing by the host of the Command register in the device with a command code value. The Command object is instantiated in the serial ATA transport protocol by the command code value placed in the Command Frame Information structure by the host. These objects exhibit well-defined and observable behaviors, but they do not exist as separate physical elements. An object may be a single numeric parameter (e.g., a tag) or a complex entity that performs a set of operations or services on behalf of another object. Other objects are required to understand some ATA functions but have implementation definitions outside the scope of this standard (e.g., the task file registers).

Service interfaces are defined between distributed objects and protocol layers. The template for a distributed service interface is the client-server model described in 4.4. The structure of an ATA I/O system is specified in 4.2 by defining the relationship among objects. The set of distributed services to be provided are specified in 4.3.

Requirements that apply to each ATA transport protocol standard are specified in the ATA command and device management model described in clause 5. The model describes required behavior in terms of layers, objects within layers and ATA transport protocol service transactions between layers.

4.2 ATA structural model

4.2.1 The ATA domain

The fundamental object of the ATA structural model is the ATA domain that represents an I/O system. A domain is made up of one host, one or more devices, and a service delivery subsystem that transports commands, data, device management functions, and related information. Figure 3 shows a schematic of an example of a simple ATA domain.

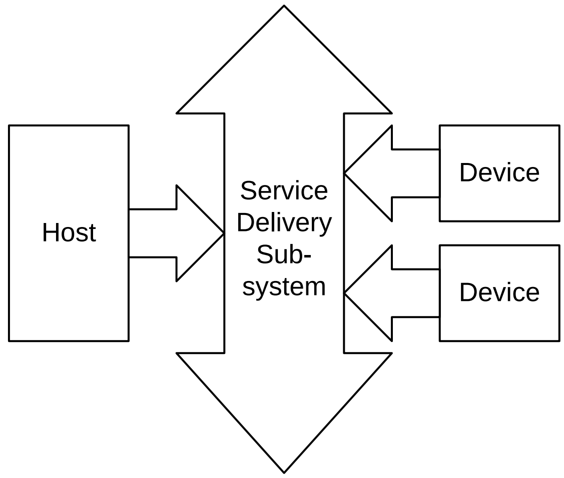

Figure 3 — Simple ATA domain example

[editor's note: there are other objects that could be in an ATA domain, including SATA port selectors, SATA port multipliers, and Serial ATA enclosure management bridges. These should be included in AAM at some point in time, but, assuming we can include these objects, I want to get a first pass at everything else first.]

It is possible for a device to be connected though a bridging or translating component to an infrastructure containing other components. In these cases, because the device only comprehends one host, the entire infrastructure, including the bridging or translating component is considered by this standard to be the host. For example, a Serial ATA device may be connected to a port in a SAS expander device that implements the Serial ATA Tunnelling Protocol (STP). Other ports in the SAS expander device may be connected to several other SAS expander or SAS initiator devices in the SAS domain, each of which may use the STP protocol to communicate with the device. In this example, the entire SAS domain is considered by this standard as the host.

4.2.2 Host

A host:

a) originates commands and device management functions;
b) transmits commands, device management functions, and data to devices; and
c) receives data and status from devices.

Traditionally, a host has been the computer system executing the software BIOS and/or operating system device drivers controlling the device and the adapter hardware for the ATA interface to the device. A host may provide other functions that are beyond the scope of this standard.

A host contains application clients and a host port. An application client is an object in a host that is the source of commands and device management requests. An application client is independent of the interconnect and ATA transport protocol (e.g., an application client may correspond to the device driver and any other code within the operating system that is capable of managing I/O requests without requiring knowledge of the interconnect or ATA transport protocol). An application client creates one or more application client tasks, each of which processes a single command or device management function. Application client tasks are part of their parent application client. The application client task directs requests to a remote server via the host port and service delivery subsystem and receives a completion response or a failure notification. The request identifies the service to be performed and includes the input data. The response conveys the output data and requests status.

A host port is an object in a host that acts as the connection between its application client(s) and the service delivery subsystem through which commands, device management functions, data, and responses are routed.

Figure 4 shows a schematic of a simple host.

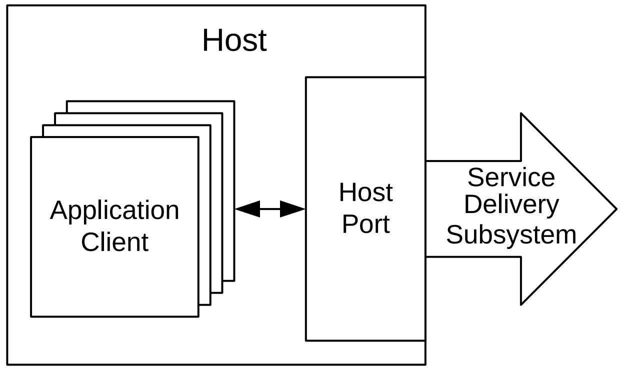

Figure 4 — Simple host schematic

4.2.3 Device

A device processes commands and device management functions, receives data from a host, and transmits data and status to a host. Traditionally, a device has been a hard disk drive, but any form of storage device may be placed on the ATA interface provided the device adheres to the set of ATA standards.

A device contains a device port, a device manager, and a device server. The device port is the object in a device that acts as the connection between its device server and device manager and the service delivery subsystem through which commands, device management functions, data, and status are routed. The device manager is the object in a device that controls the sequencing of one or more commands and processes device management functions. The device server is the object in a device that processes commands that transfer data. The objects in a device may operate concurrently (e.g., a device port may be receiving a command while a device server is processing data for a previous command).

Figure 5 shows a schematic of a simple device.

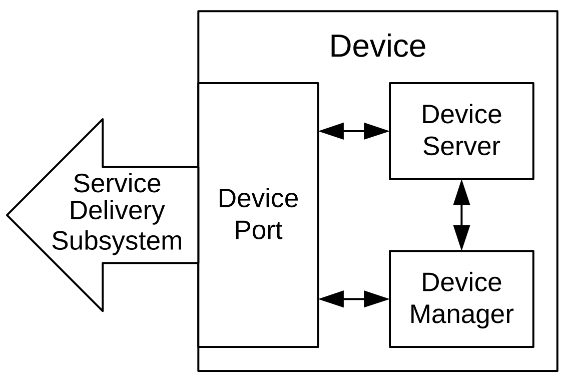

Figure 5 — Simple device schematic

4.2.4 Service delivery subsystem

The service delivery subsystem connects host ports and device ports and is a single path for the transfer of requests and responses between a host and a device. The service delivery subsystem is assumed to transport error-free copies of the requests and responses between sender and receiver.

Figure 6 shows a schematic of an example ATA system including all of its components.

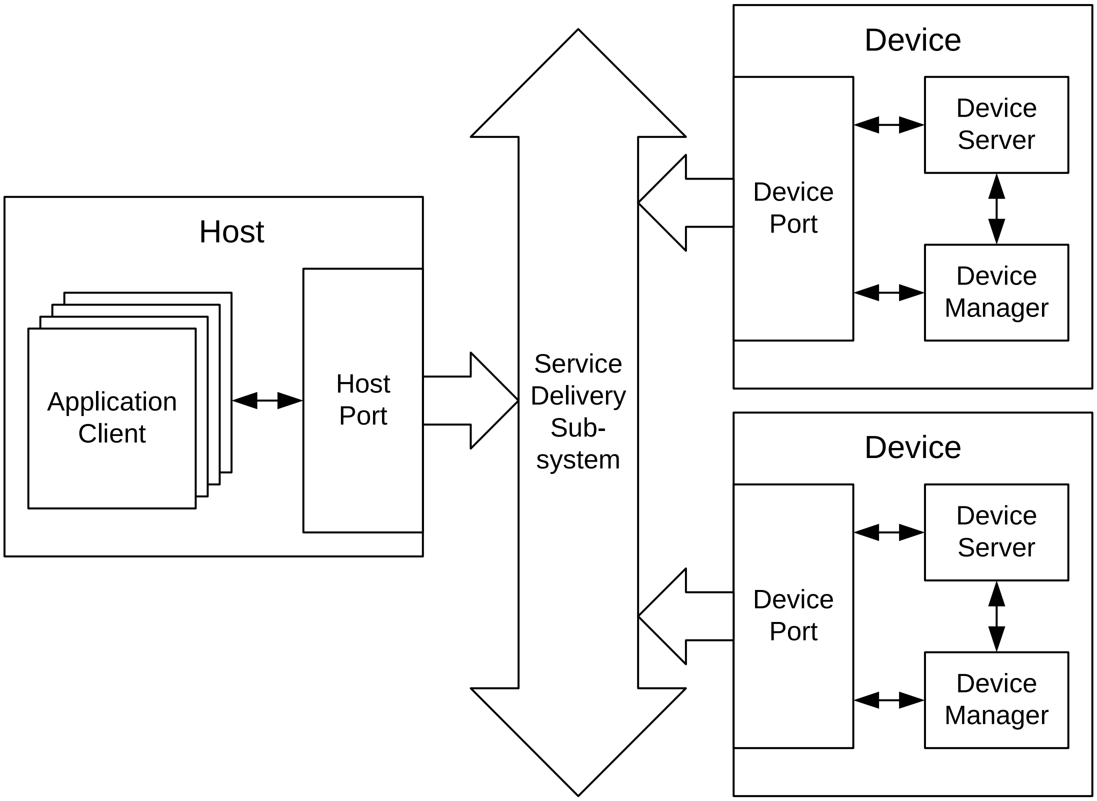

Figure 6 — Example ATA system schematic

4.3 The ATA distributed service model

Transactions between distributed objects are represented in this standard by the client-server model. A client-server transaction is represented as a procedure call with inputs supplied by the caller (i.e., the application client). The procedure call is processed by the device server or device manager and returns outputs and a procedure call status.

4.4 The ATA client-server model

All requests originate from the application client residing within a host. An application client creates an application client task to process a request. Each request takes the form of a procedure call with arguments and a status to be returned. An application client task requests processing of a command to transfer data through a request directed to the device server within a device. An application client task requests processing of a task management function through a request directed to the device manager within a device. An application client task ceases to exist once the command or device management function ends (i.e., status for the command or task management functions has been returned to the application client).

As seen by the application client, a request becomes pending when it is passed to the host port for transmission. The request is complete when the server response is received or when a device management function is sent (e.g., a reset). As seen by the device server, the request becomes pending upon receipt and completes when the response is passed to the device port for return to the application client. As a result there may be a time skew between the device servers and application clients perception of request status and server state. All references to a pending command or device management function in this standard are from the application clients perspective.

Although a device driver in an ATA implementation may perform transfers through several interactions with its transport protocol layer, the architecture model portrays each operation, from the viewpoint of the application client, as occurring in one discrete step. The request or response is:

a) Considered sent when the sender passes it to the source port for transmission;
b) In transit until delivered to the receiving port and any handshaking occurs; and
c) Considered received when it has been forwarded to the receiver via the destination port.

Client-server relationships are not symmetrical. An application client may only originate requests for service. A device server may only respond to such requests. The application client requests an operation provided by a device server located in a device and waits for completion, which includes transmission of the request to and response from the remote server. In this model, confirmation of successful request or response delivery by the sender is not required. The model assumes that delivery failures are detected by the host port or within the service delivery subsystem.

5 ATA command and device management model

5.1 ATA command and device management model overview

There are four ATA procedure calls:

a) The Execute Command procedure call;
b) The Data-In Delivery procedure call;
c) The Data-Out Delivery procedure call; and
d) The Device Management procedure call.

Each of the procedure calls invokes transport services. Each transport service contains one or more of the following arguments:

a) The Device argument specifies the device selected to execute the procedure call;
b) The Command argument specifies the code identifying the unit of work to be performed by the device;
c) The LBA argument specifies:
    A) the LBA of the data to be transferred if only one logical block is to be transferred;
    B) the first LBA in a consecutive sequence of LBAs if more than one logical block is to be transferred; or
    C) command-specific information (e.g., the LBA information argument for SMART commands contains a value unique to that command set);
d) The Sector Count argument specifies the number of logical blocks to be transferred;
e) The Byte Count argument specifies the number of bytes to be transferred;
f) The Features argument specifies additional command-specific information;
g) The Control argument specifies actions a device management function a device is to execute;
h) The Host Buffer argument contains the beginning address of the buffer in the host to where or from where data is to be transferred for a Data-In or Data-Out transaction sequence;
i) The Device Buffer argument contains the beginning address of the buffer in the device from where or to where data is to be transferred for a Data-In or Data-Out transaction sequence;
j) The Transfer Count argument specifies or indicates the amount of data transferred for a particular transfer;
k) The Tag argument specifies or indicates a number assigned to a particular Data-In or Data-Out transaction sequence;
l) The Status argument indicates information about command completion and/or condition of a device; and
m) The Error argument indicates additional information if a command completion has resulted in an error.

Other transport-specific arguments not described in this standard may be included in a transport service.

The format used in this standard to describe each transport service is:

Name type1 type2 (argument 1, [argument 2])

Where:

a) Name is the name of the transport service being invoked. The transport services are:
    A) Send Command protocol service request (see 5.2);
    B) Command Received protocol service indication (see 5.2);
    C) Send Command Complete protocol service response (see 5.2);
    D) Command Complete Received protocol service confirmation (see 5.2);
    E) Send Data-In data transfer service request (see 5.3);
    F) Data-In Delivered data transfer service confirmation (see 5.3);
    G) Receive Data-Out data transfer service request (see 5.4);
    H) Data-Out Received data transfer service confirmation (see 5.4);
    I) Send Device Management protocol service request (see 5.5);
    J) Device Management Request Received protocol service indication (see 5.5);
    K) Device Management Function Executed protocol service response (see 5.5); and
    L) Received Device Management Function Executed protocol service confirmation (see 5.5);
b) Type 1 is either “protocol service” or “data transfer service”;
c) Type 2 is “request”, ”response”, “indication”, or “confirmation”;

d) (argument 1) is one or more arguments that are mandatory for the transport service; and
e) [argument 2] is an argument that is optional or specific to a transport protocol.

An example of a transport service as used in this standard is: Send Command protocol service request ([Device], Command, [LBA], [Sector Count], [Byte Count], [Features], [Host Buffer], [Tag]).

5.2 The Execute Command procedure call

The application client requests the processing of a command by invoking the Execute Command procedure call. There are four transport protocol services that support the procedure call. In order of execution the services are:

1) Send Command protocol service request ([Device], Command, [LBA], [Sector Count], [Byte Count], [Features], [Host Buffer], [Tag]) is sent from the application client to the host port to notify the host port that it is to begin a transaction sequence;
2) Command Received protocol service indication (Command, [LBA], [Sector Count], [Byte Count], [Features], [Tag]) is sent from the device port through the device manager to notify the device server that the host is requesting a transaction sequence;
3) Send Command Complete protocol service response (Status, [Tag], [Error]) is sent from the device server through the device manager to notify the device port that a transaction is complete and a response is to be sent to the host; and
4) Command Complete Received protocol service confirmation ([Device], Status, [Tag], [Error]) is sent from the host port to notify the application client that a transaction sequence is complete and response has been received from the device.

Figure 7 shows the protocol services of the Execute Command procedure call.

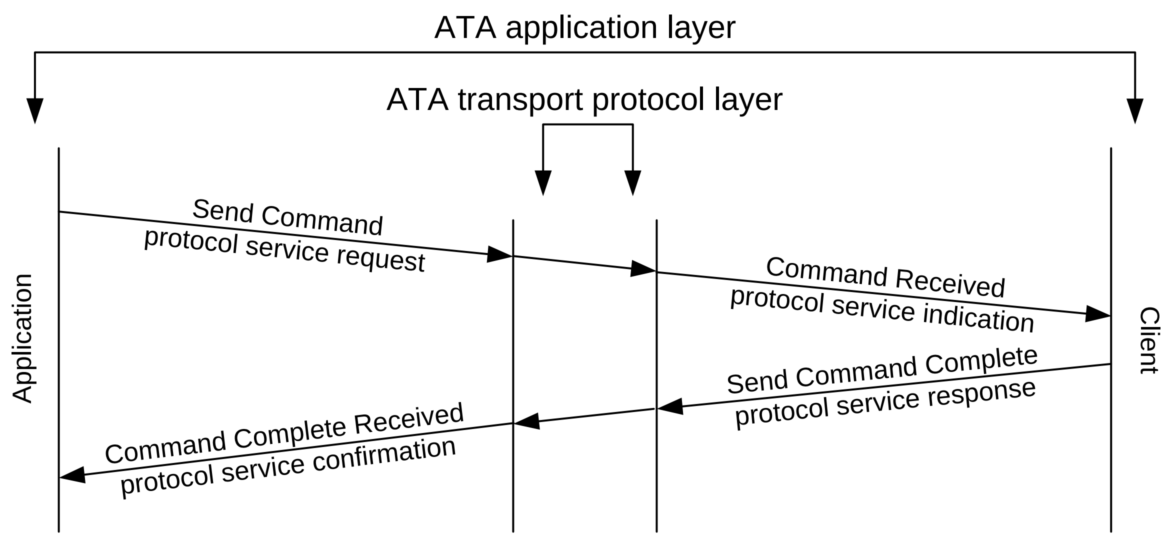

Figure 7 — Execute Command procedure call

5.3 The Data-In Delivery procedure call

The device server requests that the device port transfer data to the host port in response to an Execute Command procedure call requesting read data by invoking the Data-In Delivery procedure call. There are two data transport services that support the procedure call. In order of execution the services are:

1) Send Data-In data transfer service request ([Tag], Device Buffer, [Host Buffer], Transfer Count) is sent from the device server to notify the device port that data-in data is ready for transfer; and
2) Data-In Delivered data transfer service confirmation ([Tag], Status, [Error]) is sent from the device port to notify the device server that data-in data was successfully delivered to the host or that a service delivery subsystem error occurred while attempting to deliver the data.

Figure 8 shows the protocol services of the Data-In Delivery procedure call.

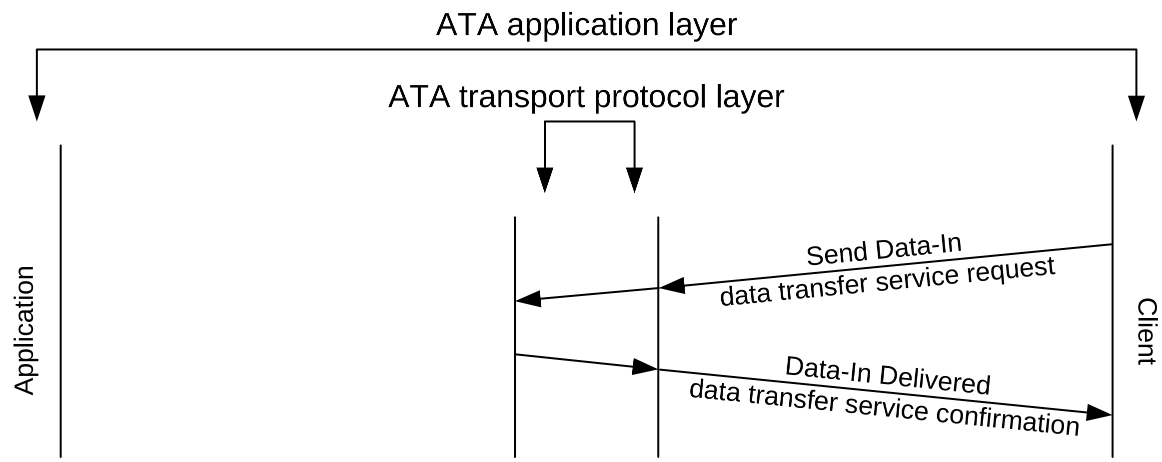

Figure 8 — Data-In Delivery procedure call

5.4 The Data-out Delivery procedure call

The device server requests that the device port receive write data from the host port in response to an Execute Command procedure call by invoking the Data-Out Delivery procedure call. There are two data transport services that support the procedure call. In order of execution the services are:

1) Receive Data-Out data transfer service request ([Tag], Device Buffer, [Host Buffer], Transfer Count) is sent from the device server to notify the device port that the host will be sending data-out data; and
2) Data-Out Received data transfer service confirmation ([Tag], Status, [Error]) is sent from the device port to notify the device server that data-out data was successfully received from the host or that a service delivery subsystem error occurred while attempting to deliver the data.

Figure 9 shows the protocol services of the Data-Out Delivery procedure call.

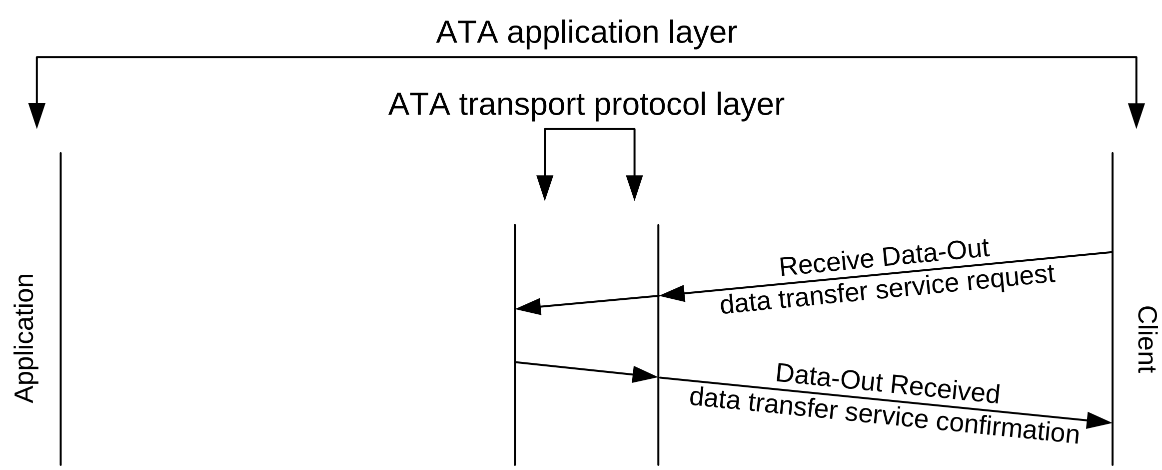

Figure 9 — Data-Out delivery procedure call

[editor's note: should there be separate procedures for PIO and DMA? The differences are detailed in the protocol documents and AAM could be agnostic. However, there are enough similarities (e.g., size of DRQ data block and when status is reported) that I could try to pull out the common bits and make separate procedures.]

5.5 The Device Management procedure call

The application client requests that a device manager process a device management function by invoking the Device Management procedure call. There are four transport protocol services that support the procedure call. In order of execution the services are:

1) Send Device Management protocol service request ([Device], Control) is sent from the application client to the host port to notify the host port that it is to begin a Device Management transaction sequence;
2) Device Management Request Received protocol service indication (Control) is sent from the device port to notify the device manager that it is to perform a device management function;
3) Device Management Function Executed protocol service response ([Device], Status, [Error]) is sent from the device manager to notify the device port that a device management function is complete; and
4) Received Device Management Function Executed protocol service confirmation ([Device], Status, [Error]) is sent from the host port to notify the application client that a device management function is complete.

Figure 10 shows the protocol services of the Device Management procedure call.

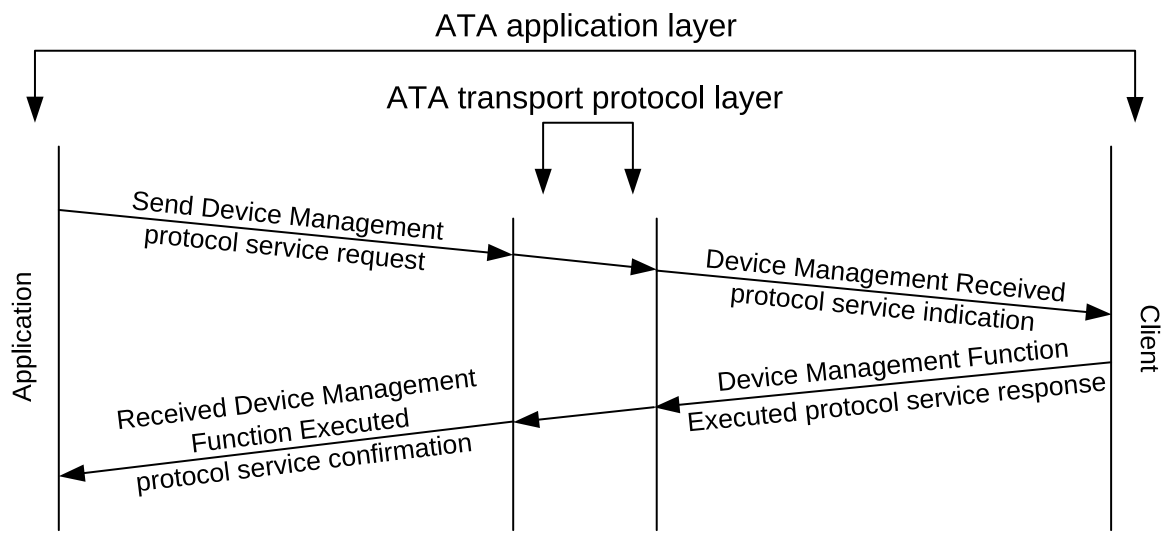

Figure 10 — Device Management procedure call

6 ATA events and event notification model

6.1 ATA events overview

ATA events may occur or be detected in one of the following:

a) The device port;
a) The host port;
b) The device server;
c) The device manager; or
d) An application client.

The detection of any event may require processing by the object that detects it.

Events that occur in a device are assumed to be detected and processed by all objects within the device. Events that occur in a host are assumed to be detected and processed by all objects within the host.