Working Draft
American National
Standard

Project
T13/1700-D

Revision 3
25 May 2006

Information technology – AT Attachment – 8 ATA/ATAPI Architecture Model (ATA8-AAM)

T13 Technical Editors:

Mark Overby
nVidiaCorporation
1430 NE 24th Suite 203
Bellvue, WA 98007
USA

Telephone: 425-417-9412
Email: moverby@nvidia.com

Mark Evans
Western Digital Corporation
5863 Rue Ferrari
San Jose, CA 95138

Email: mark.evans@wdc.com
Office: 408.363.5257

Reference number
ISO/IEC 14776-861:200x
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

T13 Vice-chair

James Hatfield
Seagate Technology
389 Disc Drive
Longmont, CO 80503
USA

Telephone: 720-684-2120

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 – 8 ATA/ATAPI Architecture Model (ATA8-AAM)

Secretariat

Information Technology Industry Council

Approved: mm/dd/yyyy

American National Standards Institute, Inc.

Abstract

This standard describes 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 define those aspects of ATA system behavior that are independent of a particular technology and common to all implementations.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard

Published by:
American National Standards Institute
11 West 42nd Street, New York, New York 10036

Printed in the United States of America

Revision History

R.1 Revision ATA8-AAMr00 (22 February 2005)

First release of ATA8-AM.

R.2 Revision ATA8-AAMr0a (20 June 2005)

a) Updated the names of the ATA-8 standards throughout the document;
b) Updated the figure titled, “Standards and specifications relationships”, per the working group input, and added descriptions for the new objects in the figure;
c) Updated the figure titled, “Example ATA system schematic”, and changed its name to “Example ATA domain with two devices”;
d) Removed the Device Manager object and moved its functionality to the Device Server object;
e) Deleted the Sector Count and Byte Count arguments and added the Count argument;
f) Added the DRQ Data Block argument;
g) Removed the description of the Execute Command, Data-In Delivery, Data-Out Delivery, and Device Management procedure calls;
h) Added the ATA protocols and began describing them as procedure calls using the transport services; and
i) Added clause 6 to describe ATA events (e.g., resets).

R.3 Revision ATA8-AAMr0b (22 August 2005)

a) Included input from the T13 working group, Thursday, June 23, 2005;
b) Added the DMA queued command protocol;
c) Added the Packet command protocol;
d) Added sequence numbers to the command protocol figures
e) Added power-on and device management protocols;
f) Deleted the events clause as all events are included in the protocols;
g) Deleted unused references;
h) Removed the lists of commands for each protocol as these are contained in ATA8-ACS;
i) Expanded clause 4 to include the description of layering; and
j) Made other editorial corrections and changes.

R.4 Revision ATA8-AAMr1 (09 September 2005)

Note that several of the following changes (e.g., moving references standards to a table) are not marked in this revision as a change from the previous revision, as these changes are editorial.

a) Included input from George Penokie;
b) Included input from line-by-line review conducted at the T13 working group, Thursday, August 25, 2005, including:
    A) Deleted unused references;
    B) Added used references;
    C) Placed references in tables;
    D) List notation format was expanded
    E) The Packet command protocol was expanded to include one each for non-data, PIO data-in, PIO data-out, and DMA;
    F) Updated all protocol figures to be consistent; and
    G) Many minor editorial changes.

One version of revision 1 was made indicating changes from revision 0b. The file name for this version is D1700r1-ATA8-AAM with changes from r01b. In this version, some minor editorial changes from revision 0b to revision 1 are not marked. Some significant changes from revision 0b to revision 1 are not marked but are

indicated by an editor's note. A second version of revision 1 was made in which all indications of change from revision 0b were removed. The file name for this version is D1700r1-ATA8-AAM.

R.5 Revision ATA8-AAMr2 (04 November 2005)

a) Changed "command and device management model" to "protocol model";
b) Added "nexus loss event" in the definitions;
c) Changed "Transport Reset Received" confirmation to "Transport Event Notification Received" indication;
d) Added the Nexus Loss protocol service argument;
e) Added the nexus loss protocol;
f) Included other input from Rob Elliott;
g) Included input from line-by-line review conducted at the T13 working group, Thursday, October 20, 2005, including:
   A) Updating all referenced standards and specifications to the latest information available (these changes are not marked);
   B) Added a table showing which arguments are used for which transport protocol services for which protocol (this change is not marked);
   C) Changed the format of the transport protocol services from "Name type (argument 1, [argument 2])" to "Name (argument 1, [argument 2]) type" throughout clause 5 (these changes are not marked);
   D) Split the power-on protocol into two protocols, one for the host and one for the device (the text deleted from what became the host power-on protocol was not marked as a change);
   E) Attempted to make the data transfer protocol figures more readable;
   F) Added the LBA argument to many transport protocol services;
   and
h) After making the changes itemized in this list, made a revision of the draft for letter ballot removing the change markings.

R.6 Revision ATA8-AAMr2a (27 January 2006)

a) Included resolutions to all comments marked "accepted" in proposal e05-185r0_ATA8-ACS_letter_ballot_comment_resolution; and
b) Made several editorial changes to simplify and improve clarity and consistency.

Note: Because the changes were so extensive, particularly in clause 5, changes are marked with change bars only.

R.7 Revision ATA8-AAMr2b (24 March 2006)

a) Changed the document name throughout to be: AT Attachment – 8 ATA/ATAPI Architecture Model (ATA8-AAM);
b) Included resolutions to comments from the February working group meeting (see proposal e05-185r1_ATA8-ACS_letter_ballot_comment_resolution). The most significant change was the addition of the Interrupt argument to each Send Command Function Complete protocol service response; and
c) added definitions for power-on, hardware, and software events and resets.

Note: all changes from revision 2a are marked with change bars, except in the table of contents, table of tables, and table of figures. Only some of the more significant changes are marked with blue underlined text and red strike through text in other parts of the draft.

R.8 Revision ATA8-AAMr2c (03 April 2006)

Expanded the definition for resets showing which is included where.

R.9 Revision ATA8-AAMr3 (25 May 2006)

Removed all change markings from revision 2c.

Contents

Page

1 Scope 1
1.1 Introduction 1
1.2 Requirements precedence 1
1.3 ATA family of standards 2

2 References 4
2.1 References overview 4
2.2 Normative references 4
2.2.1 Normative references overview 4
2.2.2 Approved normative references 4
2.2.3 Normative references under development 5
2.3 Other references 5

3 Definitions, symbols, abbreviations, and conventions 6
3.1 Definitions 6
3.2 Symbols 7
3.3 Abbreviations 7
3.4 Keywords 7
3.5 Conventions 8
3.5.1 Editorial conventions 8
3.5.2 Numeric conventions 8
3.5.3 Lists 8
3.5.3.1 Lists overview 8
3.5.3.2 Unordered lists 9
3.5.3.3 Ordered lists 9
3.5.4 Precedence 9

4 ATA architecture model 10
4.1 Introduction 10
4.2 ATA structural model 10
4.2.1 The ATA domain 10
4.2.2 The host 11
4.2.3 The device 12
4.2.4 Service delivery subsystem 12
4.3 ATA architecture layering 14
4.4 The ATA client-server model 15
4.5 The ATA distributed service model 16
4.5.1 The ATA distributed service model overview 16
4.5.2 Transport protocol services 16
4.5.3 Data transfer protocol services 17

5 ATA protocol model 18
5.1 ATA protocol introduction 18
5.1.1 ATA protocol overview 18
5.1.2 ATA protocol services 19
5.1.3 ATA protocol service arguments 20
5.1.4 ATA transport service format 22
5.2 Power-on, nexus loss, and device management protocols 22
5.2.1 Power-on protocols 22
5.2.1.1 Power-on protocols overview 22
5.2.1.2 Host Power-on protocol description 23
5.2.1.3 Device Power-on protocol description 23
5.2.2 Nexus Loss protocol 24
5.2.2.1 Nexus Loss protocol overview 24

5.2.2.2 Nexus Loss protocol description 24
5.2.3 Device Management protocol 24
5.2.3.1 Device Management protocol overview 24
5.2.3.2 Device Management protocol description 25
5.3 Command protocols 26
5.3.1 Command protocol overview 26
5.3.2 Non-data Command protocol 26
5.3.2.1 Non-data Command protocol overview 26
5.3.2.2 Non-data Command protocol description 26
5.3.3 PIO Data-In Command protocol 27
5.3.3.1 PIO Data-In Command protocol overview 27
5.3.3.2 PIO Data-In Command protocol description 27
5.3.4 PIO Data-Out Command protocol 28
5.3.4.1 PIO Data-Out Command protocol overview 28
5.3.4.2 PIO Data-Out Command protocol description 29
5.3.5 DMA command protocol 30
5.3.5.1 DMA Command protocol overview 30
5.3.5.2 DMA Command protocol description 30
5.3.6 DMA Queued Command protocol 31
5.3.6.1 DMA Queued Command protocol overview 31
5.3.6.2 DMA Queued Command protocol description 32
5.3.7 PACKET Command protocol 33
5.3.7.1 PACKET Command protocol overview 33
5.3.7.2 PACKET Command protocol description 34
5.3.7.2.1 PACKET command transfer 34
5.3.7.2.2 PACKET command completion with no data transfer 35
5.3.7.2.3 PACKET command completion with read data transfer using PIO 35
5.3.7.2.4 PACKET command completion with write data transfer using PIO 36
5.3.7.2.5 PACKET command completion with data transfer using DMA 37

Tables

Page

1 Related host standards examples 2
2 Packet delivered command set standards and specifications examples 3
3 Other related device specifications examples 3
4 Standards bodies 4
5 Approved normative references 4
6 Normative references under development 5
7 Other specifications 5
8 American and ISO numbering conventions 8
9 ATA protocol services 19
10 ATA protocol service arguments 20
11 Arguments used for protocol services 21

Figures

Page

1 Requirements precedence 1
2 Standards and specifications relationships 2
3 Simple ATA domain example 11
4 Simple host schematic 12
5 Simple device schematic 12
6 Example ATA domain with two devices 13
7 ATA architecture layering model 14
8 Client-server model 15
9 Transport protocol services 17
10 Data transfer protocol services 17
11 Host Power-on procedure call 23
12 Device Power-on procedure call 23
13 Nexus Loss procedure call 24
14 Device Management procedure call 25
15 Non-data Command procedure call 26
16 PIO Data-In Command procedure call 27
17 PIO Data-Out Command procedure call 29
18 DMA Command procedure call 30
19 DMA Queued Command procedure call 32
20 PACKET command transfer 34
21 PACKET command completion with no data transfer 35
22 PACKET command completion with read data transfer using PIO 35
23 PACKET command completion with write data transfer using PIO 36
24 PACKET command completion with data transfer using DMA 37

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 ATA 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:

Clause 1 is the scope.

Clause 2 enumerates the references that apply to this standard.

Clause 3 describes the definitions, symbols, abbreviations, and conventions used in this standard.

Clause 4 describes the overall ATA architecture model.

Clause 5 describes the ATA protocol model.

American National Standard –
For Information Technology

AT Attachment – 8 ATA/ATAPI Architecture Model (ATA8-AAM)

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 defines 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 defines 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 the requirements of this standard.

1.2 Requirements precedence

This standard defines generic requirements that pertain to ATA implementation standards and implementation requirements. An implementation requirement defines behavior in terms of measurable or observable parameters that apply to an implementation. An example of implementation requirements defined in this standard are the arguments sent with a **Send Command** transport protocol service request (see 5.1).

Generic requirements are transformed to implementation requirements by an implementation standard. An example of a generic requirement is a transport specific power-on event (see 5.2.1).

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

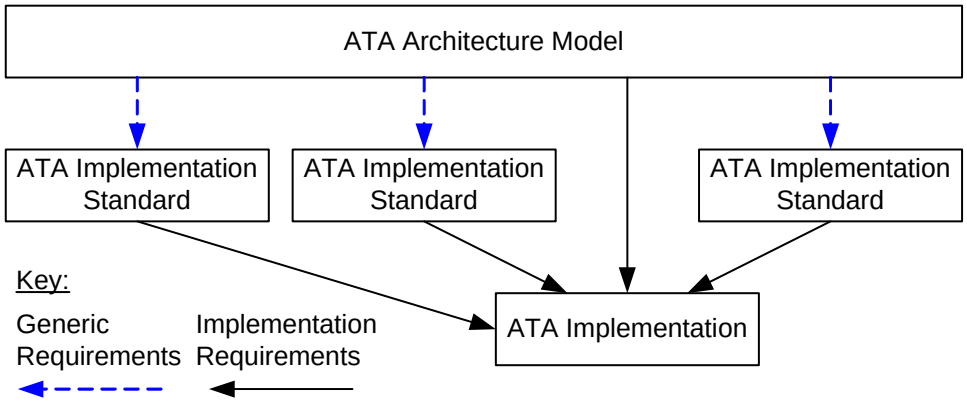

Figure 1 — Requirements precedence

As shown by the blue dotted lines in figure 1, all ATA implementation standards shall reflect the generic requirements defined in this standard. In addition, as shown by the solid black lines in figure 1, an implementation claiming compliance to this standard 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.

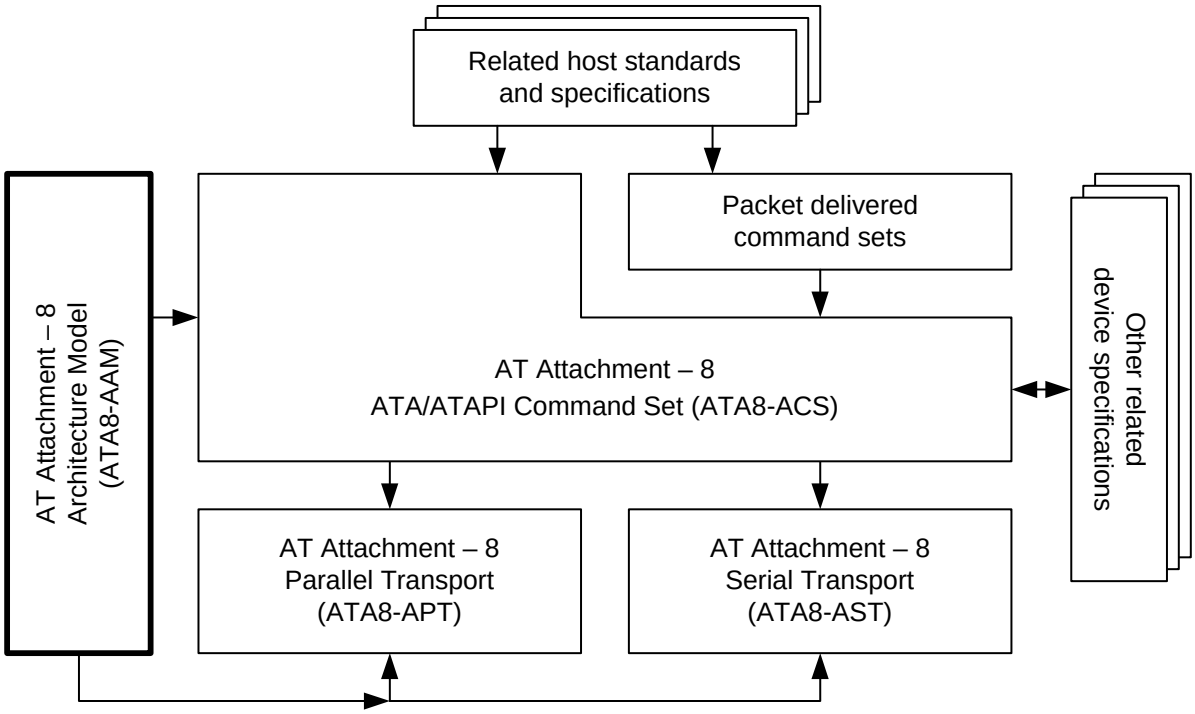

Figure 2 — Standards and specifications relationships

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:

Related host standards: define elements of host firmware and software for ATA implementations. Table 1 shows examples of related host standards.

Table 1 — Related host standards examples

Protected Area Run Time Interface Extension Services (PARTIES) standard [a]
ATA/ATAPI Host Adapter Standards (ATA Adapter) standard [a]
BIOS Enhanced Disk Drive Services – 3 (EDD-3) standard [a]
[a] See 2.2.2 for additional information about the standards listed in this table.

AT Attachment-8 Architecture Model (ATA8-AAM, this standard): 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-8 ATA/ATAPI Command Set (ATA8-ACS): defines the ATA command set used by host systems to communicate with devices (see 2.2.3 for additional information about this standard).

Packet delivered command sets: other standards and specifications define the command sets containing commands that may be delivered using the PACKET feature set (see ATA8-ACS). Table 2 shows examples of standards and specifications that define packet delivered command sets.

Table 2 — Packet delivered command set standards and specifications examples

SCSI Primary Commands – 3 (SPC-3) [a]
SCSI Block Commands – 2 (SBC-2) [a]
SCSI Multimedia Commands – 4 (MMC-4) [a]
INF-8070 ATAPI for Removable Media [b]
ATA Packet Interface (ATAPI) for Streaming Tape QIC-157 revision D [b]
[a] See 2.2.2 for additional information about these standards. [b] See 2.3 for additional information about these specifications.

Other related device specifications: define other implementations that use elements of the ATA standards. Table 3 shows examples of other related device specifications.

Table 3 — Other related device specifications examples

Serial ATA Revision 2.5 [a]
CF+ and CompactFlash™ Specification, Revision 3.0 [a]
MultiMediaCard MCA System Specification, Version 4.1 [a]
CE-ATA Digital Protocol, Revision 1.1, released 29 September 2005 [a]
[a] See 2.3 for additional information about the specifications listed in this table.

AT Attachment-8 Parallel Transport (ATA8-APT): defines the following 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; and
d) the protocols for transporting commands, data, and status using the interface.

See 2.2.3 for additional information about this standard.

AT Attachment-8 Serial Transport (ATA8-AST): defines the following 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; and
d) the protocols for transporting commands, data, and status using the interface.

See 2.2.3 for additional information about this standard.

2 References

2.1 References overview

The standards, technical reports, and specifications listed in this clause are referenced in this standard. At the time of publication, the editions indicated were valid. All standards, technical reports, and specifications 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.

2.2 Normative references

2.2.1 Normative references overview

The standards listed in this clause contain provisions that, by reference in the text, constitute provisions of this standard.

Copies of approved ANSI standards may be obtained from ANSI. For further information, contact the ANSI Customer Service Department by phone at 212-642-4900, by fax at 212-302-1286, or via the World Wide Web at http://www.ansi.org.

Table 4 shows standards bodies referenced in this standard and their web sites.

Table 4 — Standards bodies

Abbreviation	Standards body	Web site
ANSI	American National Standards Institute	http://www.ansi.org
T10	INCITS T10 SCSI storage interfaces	http://www.t10.org
T13	INCITS T13 ATA storage interface	http://www.t13.org

2.2.2 Approved normative references

At the time of publication, the referenced standards listed in table 5 had been approved.

Table 5 — Approved normative references

Name	Number
Protected Area Run Time Interface Extension Services (PARTIES) standard	ANSI INCITS 346-2001
ATA/ATAPI Host Adapter Standards (ATA Adapter) standard	ANSI INCITS 370-2004
SCSI Multimedia Commands – 4 (MMC-4) standard	ISO/IEC 14776-364, ANSI INCITS 401-2005
SCSI Block Commands – 2 (SBC-2) standard	ISO/IEC 14776-322, ANSI INCITS 405-2005
BIOS Enhanced Disk Drive Services – 3 (EDD-3) standard	ANSI INCITS 407-2005
SCSI Primary Commands – 3 (SPC-3) standard	ISO/IEC 14776-453, ANSI INCITS 408-2005

2.2.3 Normative references under development

At the time of publication, the referenced standards listed in table 6 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.

Table 6 — Normative references under development

Name	Number
AT Attachment-8 Serial Transport (ATA8-AST) draft standard	INCITS Project T13/1697-D
AT Attachment-8 Parallel Transport (ATA8-APT) draft standard	ISO/IEC 14776-881, INCITS Project T13/1698-D
AT Attachment-8 ATA/ATAPI Command Set (ATA8-ACS) draft standard	ISO/IEC 14776-871, INCITS Project T13/1699-D

2.3 Other references

Table 7 lists the other specifications referenced in this standard and where copies of the specifications may be obtained. The contact information was correct at the time of the publication of this standard

Table 7 — Other specifications

Specification	Contact information for obtaining specification
Serial ATA Revision 2.5	Serial ATA International Organization (SATA-IO) at http://www.sata-io.org
CF+ and CompactFlash™ Association Specification, Revision 3.0	CompactFlash™ Association at http://www.compactflash.org.
ATA Packet Interface (ATAPI) for Streaming Tape QIC-157 revision D	Quarter-Inch Cartridge Drive Standards, Inc., at 805 963-3853 or http://www.qic.org.
INF-8070 ATAPI for Removable Media	SFF Committee, 14426 Black Walnut Court, Saratoga, CA 95070, phone: 408-867-6630, fax: 408-867-2115, or http://www.sffcommittee.org.
MultiMediaCard MCA System Specification, Version 4.1	MultiMediaCard Association (MMCA), at http://www.mmca.org.
CE-ATA Digital Protocol, Revision 1.1	CE-ATA, at http://www.ce-ata.org

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1 application client: an object in the 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 function. (See 4.2.2.)

3.1.3 AT Attachment: the physical, electrical, transport, and command protocols for the internal attachment of storage devices to host systems as defined by the ATA family of standards.

3.1.4 ATAPI (AT Attachment Packet Interface): used by a device implementing the PACKET feature set.

3.1.5 ATA device: a device that implements the General feature set but does not implement the PACKET feature set. (See ATA8-ACS.)

3.1.6 ATA 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.)

3.1.7 ATAPI device: a device that implements the PACKET feature set but does not implement the General feature set (see ATA8-ACS).

3.1.8 command: a unit of work performed by a device that is not a device management function (e.g., a data transfer operation). (See 5.3.)

3.1.9 device: a storage peripheral that processes commands and device management functions (see 4.2.3). A device may either be an ATA device (see 3.1.5) or an ATAPI device (see 3.1.7).

3.1.10 device management function: a function performed by a device that is not a command (e.g., a reset function). (See 5.2.)

3.1.11 device port: an object in a device that acts as the connection between the device server in the device and the service delivery subsystem. (See 4.2.3.)

3.1.12 device server: an object in a device that processes commands. (See 4.2.3.)

3.1.13 DMA (direct memory access): a transport specific method of data transfer (see ATA8-APT and ATA8-AST).

3.1.14 domain: an ATA domain (see 3.1.6).

3.1.15 DRQ data block: a unit of data words associated with available status when using either the **PIO Data-In Command** protocol (see 5.3.3) or the **PIO Data-Out Command** protocol (see 5.3.4).

3.1.16 hardware reset: the routines performed by the device server and the device port in a device after a hardware reset event occurs (See 3.1.17.). The hardware reset routines performed by the device include the actions performed by the device for a software reset, and the actions defined in this standard, ATA8-ACS, and the applicable transport standards.

3.1.17 hardware reset event: occurs when a device server receives a Management Function Request Received (Hardware Reset) indication (see 5.2.3.2).

3.1.18 host: an object that originates commands and device management functions. (See 4.2.2.)

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

3.1.20 logical block: a logical sector. (See 3.1.22.)

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

3.1.22 logical sector: a set of data words accessed and referenced as a unit. A logical sector is also known as a logical block.

3.1.23 nexus loss event: a transport specific event where the host port is no longer in communication with a device port (e.g., a device was removed from a computer system). (See 5.2.2.)

3.1.24 PIO (programmed input/output): a transport specific method of data transfer (see ATA8-APT and ATA8-AST).

3.1.25 power-on event: occurs when the host or a device detects that power has been applied.

3.1.26 power-on reset: the host specific routines performed by the application client and the host port or the routines performed by the device server and the device port in a device after detecting a power-on event. The power-on reset performed by the device server and the device port include the actions performed for a hardware reset, the actions performed for a software reset, and the actions defined in this standard, ATA8-ACS, and the applicable transport standards.

3.1.27 read data: data delivered from a device port to the host port via the service delivery subsystem in response to a command.

3.1.28 service delivery subsystem: connects the host port and one or more device ports and is a single path for the transfer of requests and responses between the host and one or more devices. (See 4.2.4.)

3.1.29 software reset: the routines performed by the device server and the device port in a device after a software reset event (see 3.1.30). The software reset routines include the actions defined in this standard, ATA8-ACS, and the applicable transport standards.

3.1.30 software reset event: occurs when a device server receives a Management Function Request Received (Software Reset) indication (see 5.2.3.2).

3.1.31 write data: data delivered from the host port to a device port via the service delivery subsystem in response to a command.

3.2 Symbols

3.3 Abbreviations

ATA	AT Attachment
ATAPI	AT Attachment Packet Interface
DMA	Direct Memory Access (see 3.1.13)
I/O	Input/Output
LBA	Logical Block Address (see 3.1.21)
PIO	Programmed Input/Output (see 3.1.24)
SATA	Serial ATA

3.4 Keywords

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

3.4.2 may: a keyword that indicates flexibility of choice with no implied preference (as distinguished from the definition for “should” (see 3.4.5)).

3.4.3 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.

3.4.4 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.

3.4.5 should: a keyword indicating flexibility of choice with a strongly preferred alternative. “Should” is synonymous with the phrase “it is strongly recommended” (as distinguished from the definition for “may” (see 3.4.2)).

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).

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

3.5.2 Numeric conventions

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 8 shows some examples of decimal numbers using the American and International Standards Organization (ISO) numbering conventions.

Table 8 — 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 Lists

3.5.3.1 Lists overview

Lists shall be introduced by a complete grammatical proposition followed by a colon and completed by the items in the list.

Each item in a list shall be preceded by an identification with the style of the identification being determined by whether the list is intended to be an ordered list or an unordered list.

If the item in a list is not a complete sentence, then the first word in the item shall not be capitalized. If the item in a list is a complete sentence, then the first word in the item shall be capitalized,

Each item in a list shall end with a semicolon, except the last item, which shall end in a period. The next to the last entry in the list shall end with a semicolon followed by an “and” or an “or” (i.e., “...; and”, or “...; or”). The “and” is used if all the items in the list are required. The “or” is used if only one or more items in the list are required.

3.5.3.2 Unordered lists

An unordered list is one in which the order of the listed items in unimportant (i.e., it does not matter where in the list an item occurs as all items have equal importance). Each list item shall start with a lower case letter followed by a close parenthesis. If it is necessary to subdivide a list item further with an additional unordered list (i.e., have a nested unordered list), then the nested unordered list shall be indented and each item in the nested unordered list shall start with an upper case letter followed by a close parenthesis.

The following is an example of an unordered list with a nested unordered list:

The following are the items for the assembly:

a) a box containing:
   A) a bolt;
   B) a nut; and
   C) a washer;
b) a screwdriver; and
c) a wrench.

3.5.3.3 Ordered lists

An ordered list is one in which the order of the listed items is important (i.e., item n is required before item n+1). Each listed item starts with an Western-Arabic numeral followed by a close parenthesis. If it is necessary to subdivide a list item further with an additional unordered list (i.e., have a nested unordered list), then the nested unordered list shall be indented and each item in the nested unordered list shall start with an upper case letter followed by a close parenthesis.

The following is an example of an ordered list with a nested unordered list:

The following are the instructions for the assembly:

1) Remove the contents from the box;
2) Assemble the item;
   A) Use a screwdriver to tighten the screws; and
   B) Use a wrench to tighten the bolts;
   and
3) Take a break.

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 ATA standards may be developed;
c) provide a common reference for maintaining consistency among related ATA standards; and
d) provide the foundation for application compatibility across all ATA interconnect and transport protocol environments by defining generic requirements that apply uniformly to all ATA implementation standards within each functional area.

The development of this standard is assisted by the use of an abstract model. To describe 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 defines 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 the family of ATA standards (e.g., 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 a device with a command code value). The Command object is instantiated in the SATA transport protocol by the command code value placed in the Command Frame Information Structure (i.e., Command FIS) 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 (e.g., a device server). 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).

The structure of an ATA I/O system is described in 4.2 by defining the relationship among objects. Service interfaces are defined between distributed objects and protocol layers as described in 4.3. The template for a distributed service interface is the client-server model described in 4.4. The set of distributed services to be provided are described in 4.5.

Requirements that apply to each ATA transport protocol standard are defined 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. An ATA domain comes into existence when the host and at least one device are physically connected with each other through the service

delivery subsystem. Figure 3 shows a schematic of an example of a simple ATA domain.

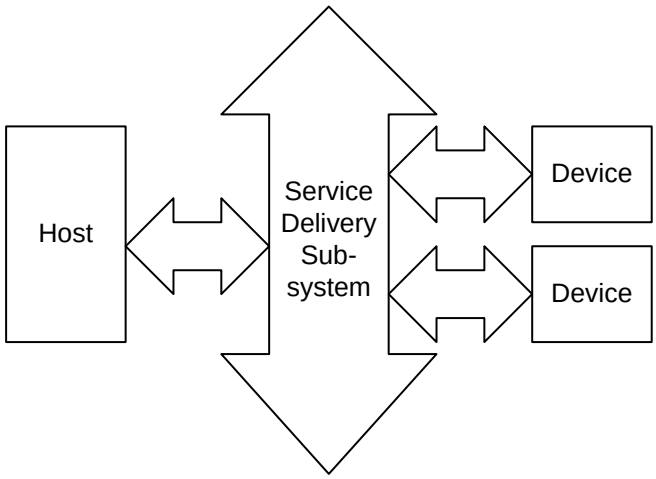

Figure 3 — Simple ATA domain example

NOTE 1 - There are other objects that may be in an ATA domain (e.g. SATA port selectors, SATA port multipliers, and SATA enclosure management bridges). These objects are outside the scope of this standard.

It is possible for a device to be connected though a service delivery subsystem to a bridging or translating component that is part of an infrastructure containing other components. In these cases, because a device only comprehends one host, the entire infrastructure, including the bridging or translating component, is considered by this standard to be the host.

4.2.2 The host

A host:

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

Traditionally, a host has been the computer system executing software, BIOS, and/or operating system device drivers controlling devices and the adapter hardware for the ATA interface to the devices. 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 (i.e., a device server (see 4.2.3)) 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 status (see 4.4).

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.

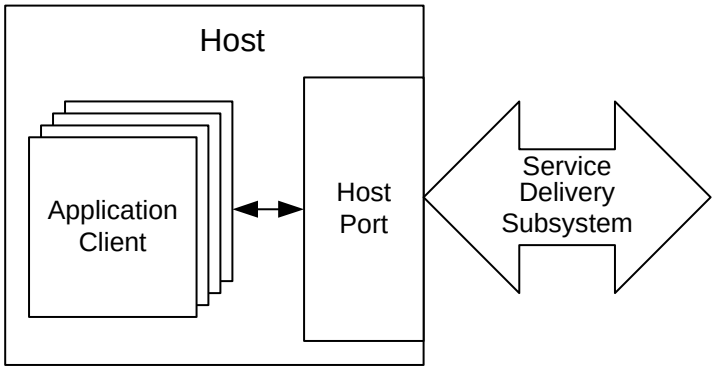

Figure 4 — Simple host schematic

4.2.3 The device

A device:

a) receives commands, device management functions, data, and other information from hosts;
b) processes commands and device management functions; and
c) transmits data, status, and other information to a host.

Traditionally, a device has been a hard disk drive, but any form of device may be placed on the ATA interface provided the device adheres to the set of ATA standards.

A device contains a device port and a device server.

The device port is the object in a device that acts as the connection between its device server and the service delivery subsystem through which commands, device management functions, data, and status are routed. The device server is the object in a device that controls the sequencing of one or more commands and processes commands and task management functions. 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.

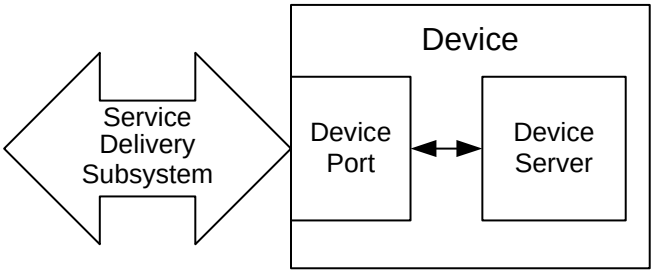

Figure 5 — Simple device schematic

4.2.4 Service delivery subsystem

The service delivery subsystem connects a host port and device ports and is a single path for the transfer of requests and responses between a host and one or more devices. For the purposes of this standard the service delivery subsystem is assumed to transport error-free copies of the requests and responses between sender and receiver.

Figure 6 shows an example of simple ATA domain with two devices including all of the objects in the domain.

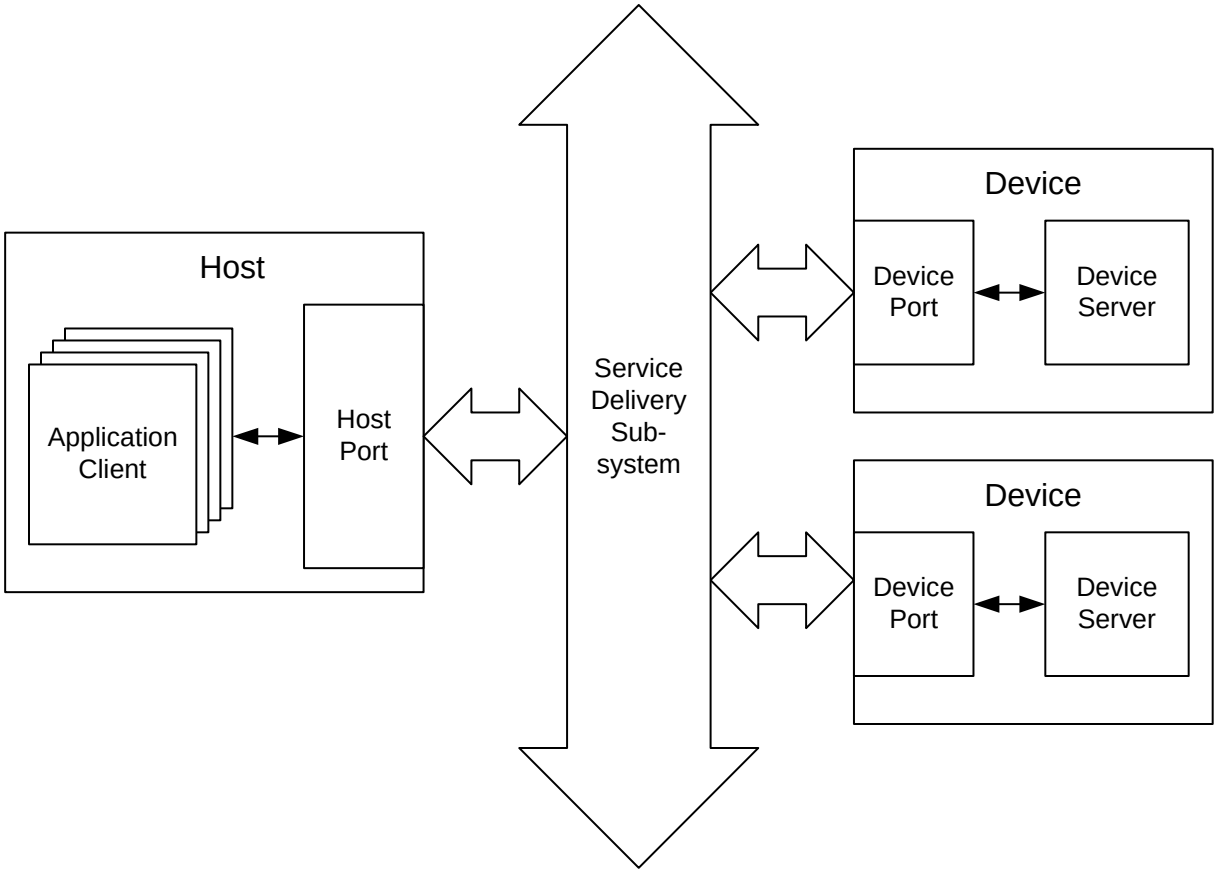

Figure 6 — Example ATA domain with two devices

4.3 ATA architecture layering

The ATA model for communications between distributed objects is based on the technique of layering. The layers of the ATA model are shown in figure 7.

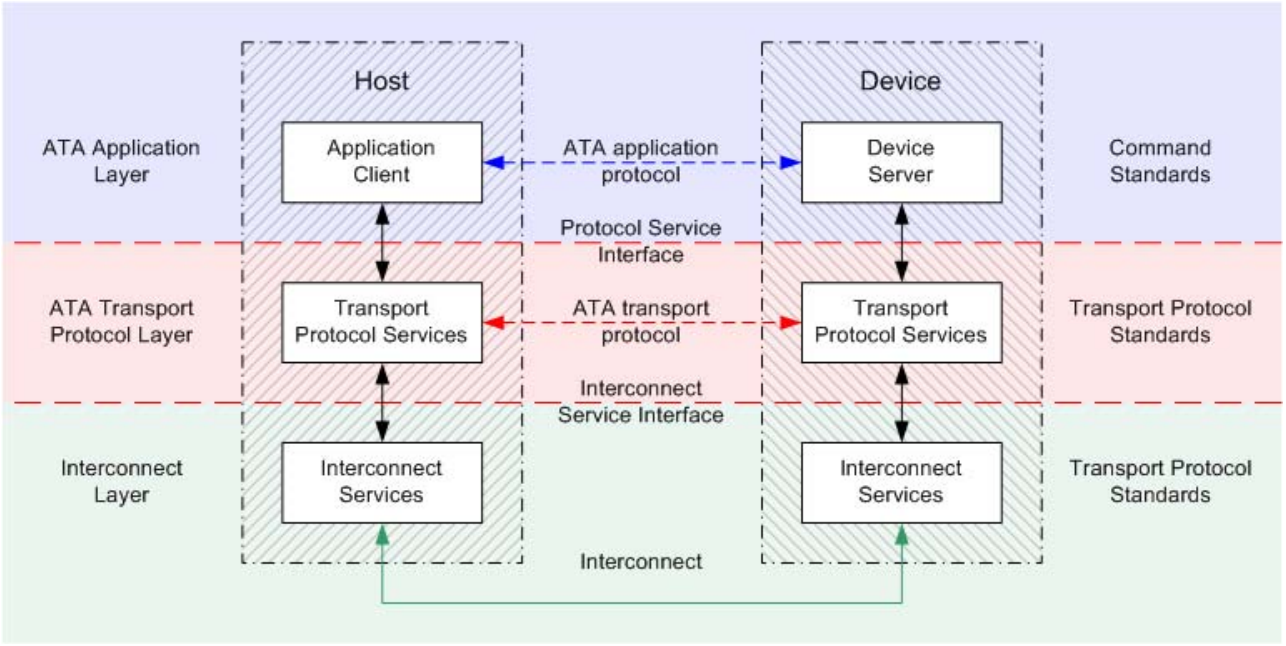

Figure 7 — ATA architecture layering model

The layers comprising this model and the specifications defining the functionality of each layer are shown in the horizontal areas in figure 7. A layer consists of peer entities in a host and device that communicate with one another by means of a protocol. Except for the interconnect layer, such communication is accomplished by invoking services provided by the adjacent layer. The following layers are defined in this model:

ATA application layer: contains the application client that originates and the device server that processes an ATA I/O operation by means of the ATA application protocol as defined in the ATA command standards and specifications (e.g., ATA8-ACS).

ATA transport protocol layer: consists of the services and protocols through which application clients communicate with device servers. The ATA transport protocol layer is contained in the host port and the device ports. The ATA transport protocol service interface is defined in this standard in representational terms using ATA transport protocol services (see 4.5). The ATA transport protocol service interface implementation is defined in each ATA transport protocol standard (e.g., ATA8-APT).

ATA interconnect layer: comprised of the services, signaling mechanisms, and the interconnect subsystem used for the physical transfer of information from sender to receiver. The ATA interconnect layer includes the physical transmission devices in the host port and the device ports, and the connectors and cabling between the ports. In the ATA model, the interconnect layer is known as the service delivery subsystem, and its components are defined in the ATA transport standards. The set of ATA transport protocol services implemented by the service delivery subsystem identify external behavioral requirements that apply to ATA transport protocol standards.

4.4 The ATA client-server model

Transactions between distributed objects are represented in this standard by the client-server model as shown in figure 8.

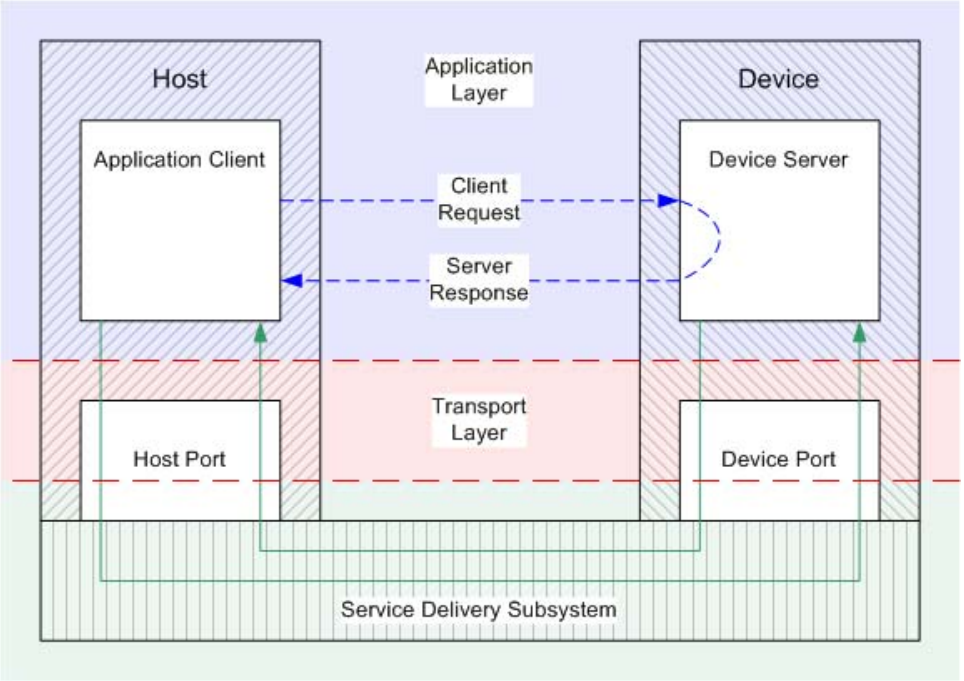

Figure 8 — Client-server model

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 recipient (i.e., the device server) and returns outputs and a procedure call status. Procedure calls are defined in this standard as protocols (see clause 5). All client requests originate from the application client residing within a host. An application client creates an application client task to process a client request. Each application client task takes the form of a procedure call with arguments. An application client task requests that a device server within a device process one of the following:

a) a non-data command;
b) a command to transfer data; or
c) a task management function.

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 client request becomes pending when it is passed to the host port for transmission. The client request is complete when the server response is received. As seen by the device server, the client request becomes pending upon receipt and completes when the server response is passed to the device port for return to the application client. As a result there may be a time skew between the application client's and the device server's perception of client request status and server response state. All references to a pending command or device management function in this standard are from the application client's 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 client request or server 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 recipient via the destination port.

Client-server relationships are not symmetrical. An application client may only originate client 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 client request to and response from the remote server. In this model, confirmation of successful client request or server 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.

4.5 The ATA distributed service model

4.5.1 The ATA distributed service model overview

Interactions between the ATA application layer and the ATA transport protocol layer are defined with respect to the application layer and may originate in either layer. An outgoing interaction is modeled as a procedure call invoking a protocol layer service. An incoming interaction is modeled as a procedure call invoked by the protocol layer.

All procedure calls may be accompanied by parameters or data. Both types of interaction are described using the notation for procedures described in this standard. Input arguments are defined relative to the layer receiving an interaction (i.e., an input is a argument supplied to the receiving layer by the layer initiating the interaction).

4.5.2 Transport protocol services

The following transport protocol service types of service interactions between the ATA application layer and the ATA transport layer are defined:

a) **Transport protocol service requests:** originate in the application layer invoking a service provided by the transport layer;
b) **Transport protocol service indications:** are sent from the transport layer, informing the application layer that an asynchronous event has occurred (e.g., the receipt of a peer-to-peer protocol transaction);
c) **Transport protocol service responses:** are sent by the application layer invoking a service provided by the transport layer to respond to a transport protocol service indication (a transport protocol service response may be invoked to return a reply from the invoking application layer to the peer application layer); and
d) **Transport protocol service confirmations:** are sent from the transport protocol layer notifying the application layer that a transport protocol service request has completed, has been terminated, or has failed to transit the interconnect layer. A confirmation may communicate parameters that indicate the completion status of the transport protocol service request or any other status. A transport protocol service confirmation may be used to convey a response from the peer application layer.

The services provided by an application layer are either confirmed or unconfirmed. An application layer service request invoking a confirmed service always results in a confirmation from the protocol layer.

Figure 9 shows an example of the relationships between the four transport protocol service types.

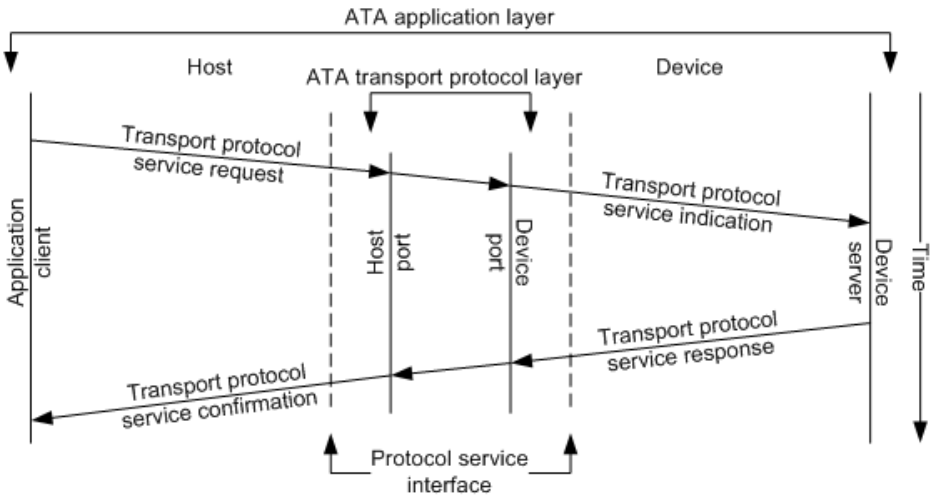

Figure 9 — Transport protocol services

4.5.3 Data transfer protocol services

When a device server invokes a data transfer protocol service, the interactions required to transfer the data do not involve the application client. Only the protocol layer in the host that contains the host port is involved.

The following data transfer protocol service types of service interactions between the ATA application layer and the ATA transport layer are defined:

a) **Data transfer protocol service requests:** originate in the application layer (i.e., the device server) invoking a service provided by the transport layer (i.e., the device port); and
b) **Data transfer service confirmations:** are sent from the transport protocol layer (i.e., the device port) notifying the application layer (i.e., the device server) that a data transfer protocol service request has completed, has been terminated, or has failed to transit the interconnect layer.

Figure 10 shows an example of the relationships between the data transfer protocol service types involved in a data transfer request.

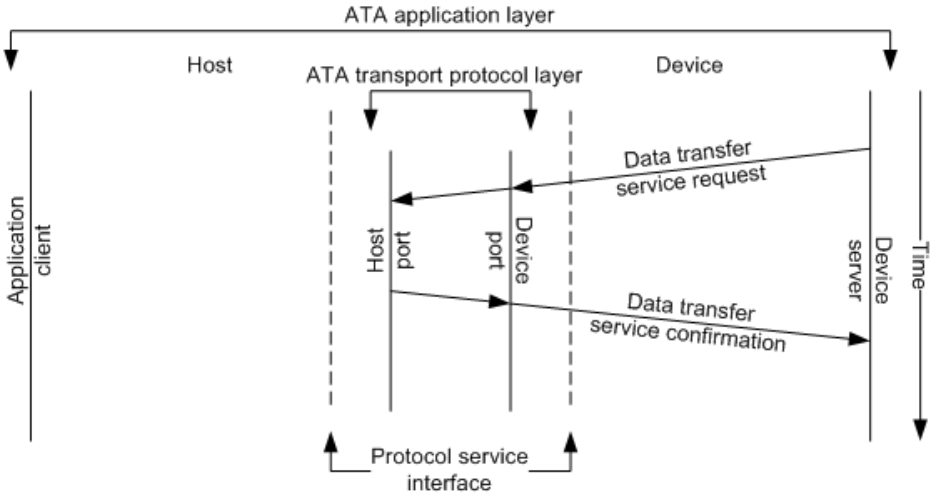

Figure 10 — Data transfer protocol services

5 ATA protocol model

5.1 ATA protocol introduction

5.1.1 ATA protocol overview

The following ATA protocol types are defined:

a) power-on, nexus loss, and device management protocols; and
b) command protocols.

The following are the power-on, nexus loss, and device management protocols:

a) **Host Power-on** protocol (see 5.2.1.2);
b) **Device Power-on** protocol (see 5.2.1.3);
c) **Nexus Loss** protocol (see 5.2.2); and
d) **Device Management** protocol (see 5.2.3).

NOTE 2 - The hardware reset protocol and software reset protocols described in ATA8-ACS, ATA8-APT, and ATA8-AST conform to the **Device Management** protocol described in this standard.

The following are the command protocols:

a) **Non-data Command** protocol (see 5.3.2);

NOTE 3 - The Execute Device Diagnostics command protocol and Device Reset command protocols described in ATA8-ACS, ATA8-APT, and ATA8-AST conform to the **Non-data Command** protocol described in this standard.

b) **PIO Data-In Command** protocol (see 5.3.3);
c) **PIO Data-Out Command** protocol (see 5.3.4);
d) **DMA Command** protocol (see 5.3.5);
e) **DMA Queued Command** protocol (see 5.3.6);
f) **PACKET Command** protocol (see 5.3.7).

Each protocol is described in this standard as a procedure call. Each ATA transport shall implement all of the protocols as they are described in this standard.

5.1.2 ATA protocol services

Each procedure call invokes a sequence of protocol services. Table 9 lists each transport protocol service and data transfer protocol service defining its type, origin, destination, and describing the notification the protocol service provides.

Table 9 — ATA protocol services

Name	Type	Sent from	Sent to	Description of the notification
Transport Event Notification Received	Transport protocol service indication	Host port	Application client	A transport specific event has occurred.
Send Management Function Request	Transport protocol service request	Application client	Host port	The host port is to deliver a device management transaction sequence request.
Management Function Request Received	Transport protocol service indication	Device port	Device server	The device server is to perform a device management function.
Send Management Function Complete	Transport protocol service response	Device server	Device port	The device port is to make a device management function completion response available.
Management Function Complete Received	Transport protocol service confirmation	Host port	Application client	A device management function is complete.
Send Command	Transport protocol service request	Application client	Host port	The host port is to deliver a command sequence request.
Command Received	Transport protocol service indication	Device port	Device server	The device server is to perform a command sequence.
Send Command Function Complete	Transport protocol service response	Device server	Device port	The device port is to make a command function completion response available.
Command Function Complete Received	Transport protocol service confirmation	Host port	Application client	A command function is complete.
Send Data-In	Data transfer protocol service request	Device server	Device port	The device port is to make read data available for delivery. Transport specific information may be sent to the host port.
Data-In Delivered	Data transfer protocol service confirmation	Device port	Device server	Read data was delivered successfully or an error occurred.
Receive Data-Out	Data transfer protocol service request	Device server	Device port	The device port is to receive write data. Transport specific information may be sent to the host port.
Data-Out Received	Data transfer protocol service confirmation	Device port	Device server	Write data was received successfully or an error occurred.

5.1.3 ATA protocol service arguments

Each transport protocol service or data transfer protocol service contains one or more arguments. Table 10 lists each argument and its definition.

Table 10 — ATA protocol service arguments

Argument	Description
Power-on	a) Indicates that an application client is to perform a power-on reset; or b) Specifies that a device server is to perform a power-on reset.
Hardware Reset	Specifies that a device server is to perform a hardware reset.
Nexus Loss	Indicates to an application client that a nexus loss event has occurred.
Software Reset	Specifies that a device server is to perform a software reset.
Enable Interrupts	Specifies that a device server is to enable its transport specific interrupt function.
Disable Interrupts	Specifies that a device server is to disable its transport specific interrupt function.
Command	Specifies the code identifying the unit of work to be performed by the device.
Device	Specifies the device selected to process the procedure call.
LBA	a) Specifies the LBA containing the data to be delivered if only one logical block is requested; b) Specifies the first LBA containing the data to be delivered in a consecutive sequence of LBAs if more than one logical block is specified; c) Specifies or indicates command specific information (e.g., for SMART commands, a value that is unique to that feature set)[a]; or d) Indicates the first LBA of an error.
Count	a) Specifies the number of logical blocks to be delivered for a command; b) Specifies the number of bytes to be delivered for a PACKET command function; or c) Specifies or indicates command specific information (e.g., the transfer mode specified in a SET FEATURES command).[a]
Features	Specifies or indicates additional command specific information.[a]
Tag	Specifies or indicates a number assigned to a particular command.
Interrupt	Indicates that other arguments in a protocol service response are valid.
Input/Output	Indicates the direction of data transfer in the **PACKET Command** protocol (see 5.3.7).[a]
Command/Data	Indicates whether a transfer is for a command or data in the **PACKET Command** protocol (see 5.3.7).[a]
DRQ Data Block	Specifies or indicates the amount of data to be delivered for transfer using PIO.
Host Buffer	Contains the beginning address of the buffer in the host to where read data or from where write data is to be delivered.
Device Buffer	Contains the beginning address of the buffer in the device from where read data or to where write data is to be delivered.
Status	Indicates command specific information about command completion and/or the condition of a device.[a]
Error	Indicates additional command specific information if a function resulted in an error.[a]

[a] Command specific information is defined in the ATA8-ACS standard.

Table 11 shows which arguments are used for which protocol services for which protocols and, when used for a protocol service for a protocol, which arguments are mandatory or optional.

Table 11 — Arguments used for protocol services

Transport protocol service / **Argument**	Power-on	Hardware Reset	Nexus Loss	Software Reset	Enable Interrupts	Disable Interrupts	Command	Device	LBA	Count	Features	Interrupt	Tag	Input/Output	Command/Data	DRQ Data Block	Host Buffer	Device Buffer	Status	Error
Transport Event Notification Received	O-M	–	L-M	–	–	–	–	L-O	–	–	–	–	–	–	–	–	–	–	–	–
Send Management Function Request		M-O	–	M-O	M-O	M-O	–	M-O	–	–	–	–	–	–	–	–	–	–	–	–
Management Function Request Received	O-M	M-O	–	M-O	M-O	M-O	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Send Management Function Complete		–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	O-M M-M	O-O M-O
Management Function Complete Received		–	–	–	–	–	–	O-O M-O	–	–	–	–	–	–	–	–	–	–	O-M M-M	O-O M-O
Send Command		–	–	–	–	–	A-M	A-O	N-O D-M Q-O	N-O D-M Q-O P-M	AC-O	–	Q-O	–	–	–	D-M Q-O	–	–	–
Command Received		–	–	–	–	–	A-M	–	N-O D-M Q-O	N-O D-M Q-O P-M	AC-O	–	Q-O	–	–	–	D-O Q-O	–	–	–
Send Command Function Complete		–	–	–	–	–	–	–	N-O D-O D-M	P-M	–	A-O	Q-O	P-M	AP-M	–	–	–	A-M	A-O
Command Function Complete Received		–	–	–	–	–	–	A-O	N-O D-O Q-O	P-M	–	–	Q-O	–	–	–	–	–	A-M	A-O
Send Data-In		–	–	–	–	–	–	–	–	D-M Q-M K-M	–	–	Q-M	–	–	PD-O	D-O Q-O K-O	D-M Q-M K-M	–	–
Data-In Delivered		–	–	–	–	–	–	–	D-O Q-O	–	–	–	Q-M	–	–	–	–	–	D-M Q-M K-M	D-O Q-O K-O
Receive Data-Out		–	–	–	–	–	–	–	–	D-M Q-M K-M	–	–	Q-M	–	–	P-M	–	D-M Q-M P-M	–	–
Data-Out Received		–	–	–	–	–	–	–	D-O Q-O	–	–	–	Q-M	–	–	–	–	–	D-M Q-M P-M	D-O Q-O P-O

Key –
– = Shall not be used for this service for any protocol
-M = Shall be used for this service for the designated protocol(s)
-O = May be used for this service for the designated protocol(s)
If an argument is not designated as either “-M” or “-O” for a protocol for a service, then the argument shall not be used for that service for that protocol

O = Power-on protocols
L = Nexus Loss protocol
M = Device Management protocol
A = All command protocols
N = Non-data Command protocol
P = PIO Data-In and/r PIO Data-Out Command protocols
D = PIO Data-In, PIO Data-Out, & DMA Command protocols
Q = DMA Queued Command protocol
P = PACKET Command protocol
K = PACKET Command protocol w/device to host xfer

Other transport specific arguments not described in this standard may be included in a transport protocol service or data transfer protocol service.

5.1.4 ATA transport service format

The format used in this standard to describe each transport protocol service or data transfer protocol service is:

Name (argument 1, [argument 2]) type

Where:

Name is the name of the transport protocol service or data transfer protocol service being invoked (see table 9).

(argument 1) represents one or more arguments that are mandatory for the transport protocol service or data transfer protocol service (see table 10).

[argument 2] represents one or more arguments that are optional for the transport protocol service or data transfer protocol service (see table 10).

type is request, indication, response, or confirmation.

An example of a transport protocol service as used in this standard is: **Send Command (Command, [Device], [LBA], [Count], [Features], [Tag], [Host Buffer])** request, where the Command argument is mandatory and the other arguments are optional or transport specific.

An example of a data transfer protocol service as used in this standard is: **Send Data-In (Count, [Host Buffer], Device Buffer)** request where the Count and Device Buffer arguments are mandatory and the Host Buffer argument is optional or transport specific.

5.2 Power-on, nexus loss, and device management protocols

5.2.1 Power-on protocols

5.2.1.1 Power-on protocols overview

Detection of a power-on event (see 3.1.25) by the host port causes the host to process its power-on protocol. Detection of a power-on event by a device port causes a device to process its power-on protocol. Because a power-on event may not be detected by the host port and a device port at the same time resulting in the host and a device processing their power-on protocols asynchronously, separate protocols are shown for the host and device.

5.2.1.2 Host Power-on protocol description

Figure 11 shows the processing of the **Host Power-on** procedure call.

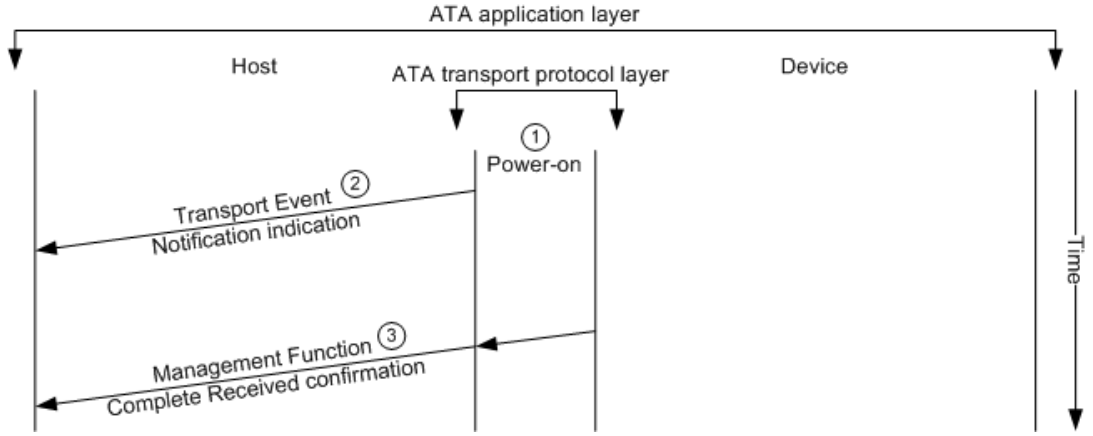

Figure 11 — Host Power-on procedure call

The following is the description of the sequence for the **Host Power-on** procedure call:

1) A power-on event occurs;
2) After detecting the power-on event, the host port performs a power-on reset and sends a **Transport Event Notification (Power-on)** indication to the application client, causing the application client to perform a power-on reset; and
3) After a response is made available to the host port by a device port (see 5.2.1.3), or a host specific event occurs, the host port sends a **Management Function Complete Received ([Device], Status, [Error])** confirmation to the application client.

It is recommended that an application client determine the current state of device specific features after completion of a power-on procedure call sequence (e.g., the host issues an IDENTIFY DEVICE command to the device, and the device makes information available that indicates the current state of the features).

The content of the arguments is defined in 5.1.3.

5.2.1.3 Device Power-on protocol description

Figure 12 shows the processing of the **Device Power-on** procedure call.

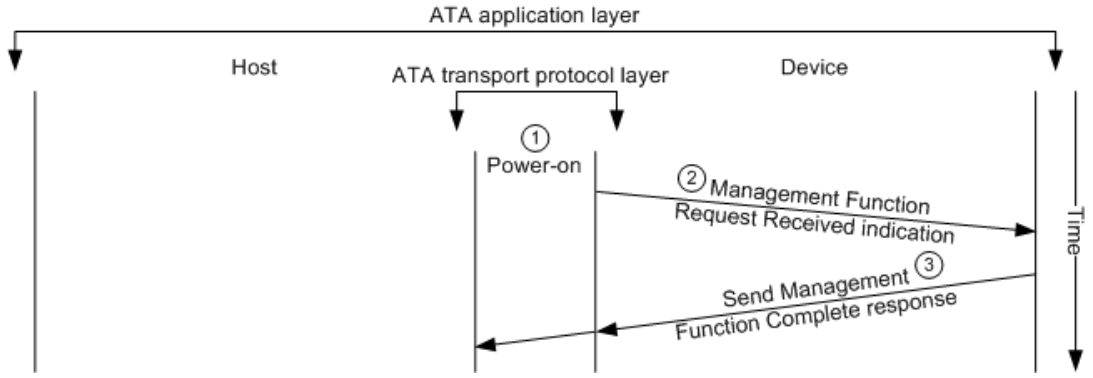

Figure 12 — Device Power-on procedure call

The following is the description of the sequence for the **Device Power-on** procedure call:

1) A power-on event occurs;

2) After detecting the power-on event, the device port performs a power-on reset and sends a **Management Function Request Received (Power-on)** indication to the device server, causing the device server to perform a power-on reset; and
3) After performing its power-on reset, the device server sends a **Send Management Function Complete (Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem.

The content of the arguments is defined in 5.1.3.

5.2.2 Nexus Loss protocol

5.2.2.1 Nexus Loss protocol overview

A transport specific nexus loss event causes a host port to notify the application client that it is no longer in communication with a device port.

5.2.2.2 Nexus Loss protocol description

Figure 13 shows the processing of the **Nexus Loss** procedure call.

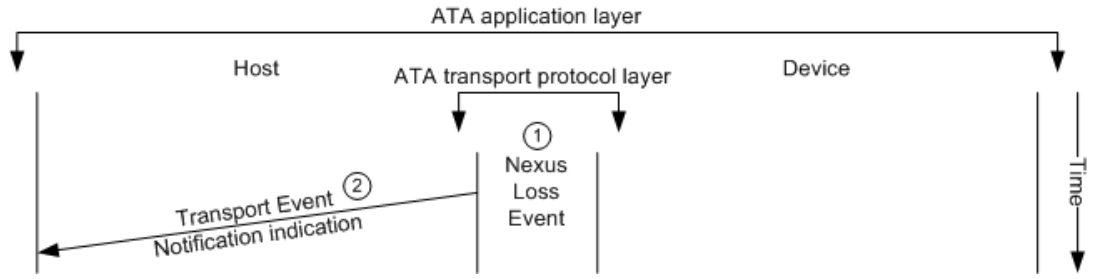

Figure 13 — Nexus Loss procedure call

The following is the description of the sequence for the **Nexus Loss** procedure call:

1) A transport specific nexus loss event occurs; and
2) After detecting the nexus loss event, the host port sends a **Transport Event Notification (Nexus Loss, [Device])** indication to the application client.

The content of the arguments is defined in 5.1.3.

5.2.3 Device Management protocol

5.2.3.1 Device Management protocol overview

A device management sequence is initiated by either the application client or the host port. Device management functions include:

a) hardware reset;
b) software reset; and
c) setting the interrupt state.

5.2.3.2 Device Management protocol description

Figure 14 shows the processing of the **Device Management** procedure call. The blue dashed lines and adjacent blue text and numbers in figure 14 show conditional or optional behavior as described in this subclause.

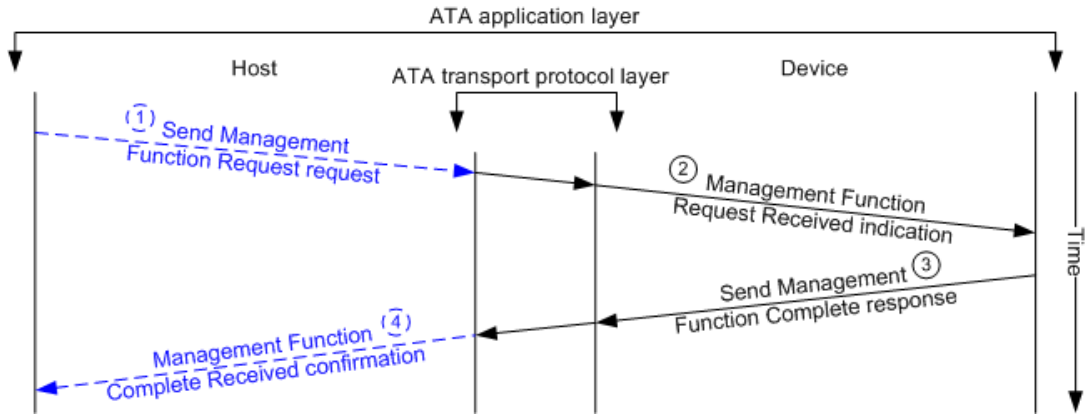

Figure 14 — Device Management procedure call

The following is the description of the sequence of the **Device Management** procedure call:

1) A device management sequence may be initiated in one of two ways:
   A) The application client sends a **Send Management Function Request ([Hardware Reset], [Software Reset], [Enable Interrupts], [Disable Interrupts], [Device])** request to the host port causing the host port to make the request available via the service delivery subsystem; or
   B) The host port generates a **Send Management Function Request ([Hardware Reset], [Software Reset], [Enable Interrupts], [Disable Interrupts])** request and makes the request available via the service delivery subsystem.

      At least one of the following arguments is included in a **Send Management Function** request:

      a) Hardware Reset;
      b) Software Reset;
      c) Enable Interrupts; or
      d) Disable Interrupts;

      No more than one of either the Power-on/Hardware Reset or the Software Reset argument is included in a **Send Management Function** request, and no more than one of either the Enable Interrupts or the Disable Interrupts argument is included in a **Send Management Function** request;

2) The device port sends a **Management Function Request Received ([Hardware Reset], [Software Reset], [Enable Interrupts], [Disable Interrupts])** indication to the device server, causing the device server to perform the requested operation:
   A) If the argument included in the **Management Function Request Received** indication is Hardware Reset, then the device server performs a hardware reset and communicates with the device port in a manner not specified in this standard causing the device port to perform a hardware reset;
   B) If the argument included in the **Management Function Request Received** indication is Software Reset, then the device server performs a software reset and communicates with the device port in a manner not specified in this standard causing the device port to perform a software reset;
   C) If the argument included in the **Management Function Request Received** indication is Enable Interrupts, then the device server enables its transport specific interrupt function; or
   D) If the argument included in the **Management Function Request Received** indication is Disable Interrupts, then the device server disables its transport specific interrupt function;
3) After performing the requested function, the device server sends a **Send Management Function Complete (Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem; and;

4) If the device management function was initiated by the application client, then, after receiving the response, the host port sends a **Management Function Complete Received ([Device], Status, [Error])** confirmation to the application client.

It is recommended that the host determine the current state of device specific features after completion of a **Device Management** procedure call if either the Hardware Reset argument or the Software Reset argument was included in the Management Function Request (e.g., the host issues an IDENTIFY DEVICE command to the device, and the device makes information available that indicates the current state of the features).

The content of the arguments is defined in 5.1.3.

5.3 Command protocols

5.3.1 Command protocol overview

Commands are grouped into different classes according to the protocol followed for command processing. The procedure calls and their constituent protocol services defined in this standard are used to describe each of the command protocols.

5.3.2 Non-data Command protocol

5.3.2.1 Non-data Command protocol overview

Processing of a **Non-data Command** procedure call involves no data transfer. See the ATA8-ACS standard for description of the non-data commands.

5.3.2.2 Non-data Command protocol description

Figure 15 shows the processing of the **Non-data Command** procedure call.

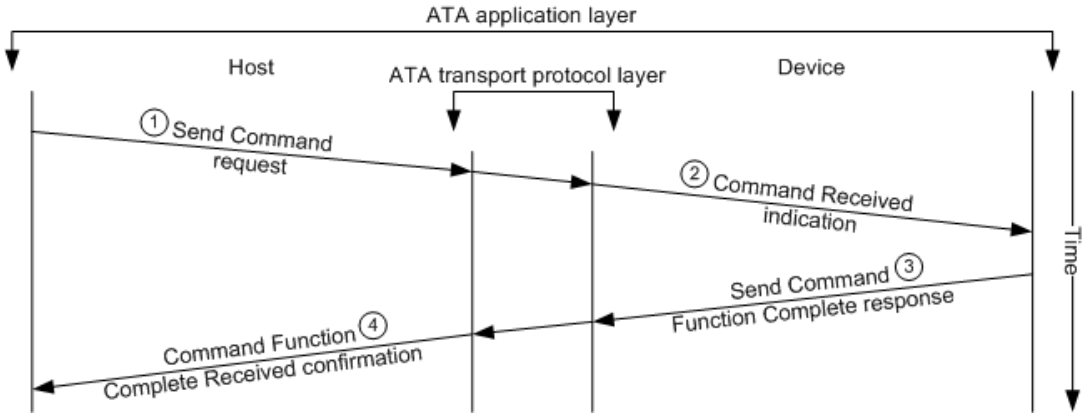

Figure 15 — Non-data Command procedure call

The following is the description of the sequence of the **Non-data Command** procedure call:

1) The application client sends a **Send Command (Command, [Device], [LBA], [Count], [Features])** request to the host port causing the host port to make the request available via the service delivery subsystem;
2) After receiving the request, the device port sends a **Command Received (Command, [LBA], [Count], [Features])** indication to the device server, causing the device server to process the command;
3) After processing the command, the device server sends a **Send Command Function Complete ([LBA], [Count], [Interrupt], Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem; and
4) After receiving the response, the host port sends a **Command Function Complete Received ([Device], [LBA], [Count], Status, [Error])** confirmation to the application client.

The content of the arguments is defined in 5.1.3.

5.3.3 PIO Data-In Command protocol

5.3.3.1 PIO Data-In Command protocol overview

Processing of a **PIO Data-In Command** procedure call includes the delivery of one or more DRQ data blocks (see 3.1.15) of read data. The **PIO Data-In Command** protocol is differentiated from the **DMA Command** protocol that transfers read data by having the device make the status available before delivering each DRQ data block, as opposed to having the device make the status available after all data for the command is delivered. See the ATA8-ACS standard for description of the PIO data-in commands.

5.3.3.2 PIO Data-In Command protocol description

Figure 16 shows the processing of the **PIO Data-In Command** procedure call. The blue dashed lines and adjacent blue text and numbers in figure 16 show conditional or optional behavior as described in this subclause.

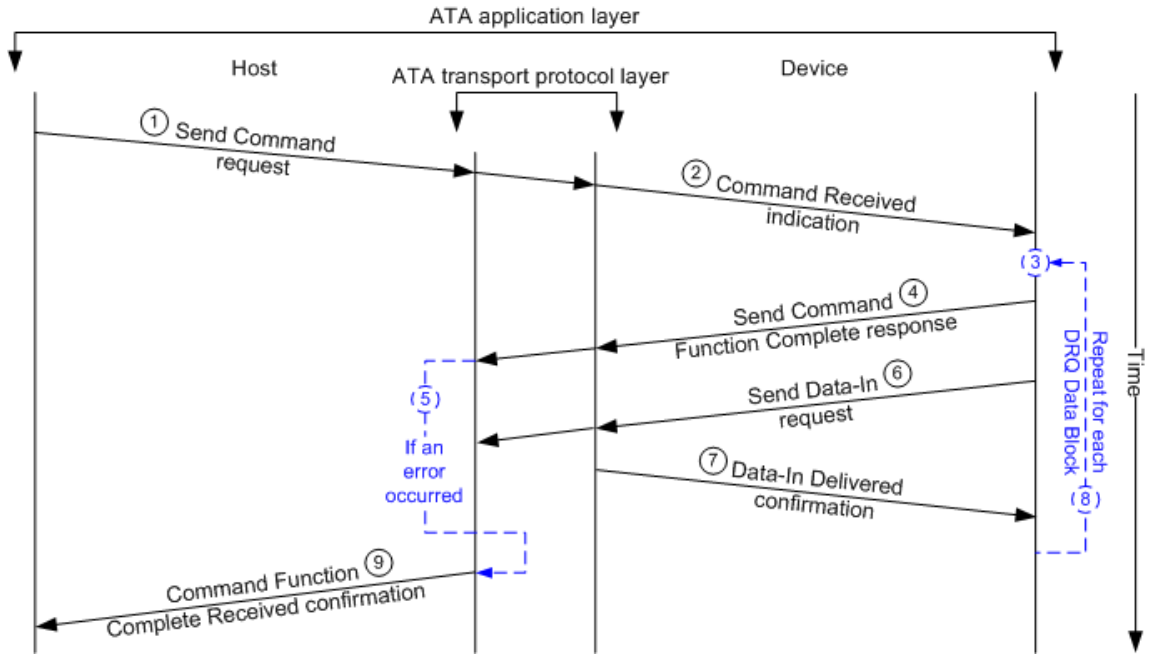

Figure 16 — PIO Data-In Command procedure call

The following is the description of the sequence of the **PIO Data-In Command** procedure call:

1) The application client sends a **Send Command (Command, [Device], LBA, Count, [Features], Host Buffer)** request to the host port causing the host port to deliver the request via the service delivery subsystem;
2) After receiving the request, the device port sends a **Command Received (Command, LBA, Count, [Features])** indication to the device server causing the device server to process the command;
3) If no error has occurred, then the device server prepares the read data for the command;
4) The device server sends a **Send Command Function Complete ([LBA], [Interrupt], Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem;
5) If the Error argument was present in the response, then the device server may stop data delivery and go to step 9;
6) The device server sends a **Send Data-In (Count, Device Buffer, DRQ Data Block)** request to the device port causing the device port to make the read data available via the service delivery subsystem.

The **Send Data-In** request may also contain transport specific information that is transmitted by the device port to the host port;

7) After all of the data has been delivered for the request, the device port sends a **Data-In Delivered ([LBA], Status, [Error])** confirmation to the device server indicating that the data was delivered or that an error occurred;
8) If there is more data to be delivered for the command, then go to step 3; and
9) After all of the data has been delivered for the command or an error has occurred, the host port sends a **Command Function Complete Received ([Device], [LBA], Status, [Error])** confirmation to the application client.

A device does not report an error if the error occurs after the **Send Command Function Complete** response for the last data for the command has been delivered but before completion of the data delivery for the command. However, there are PIO data-in commands that include a checksum in their data so the application client may verify that the data was received without error (e.g., see IDENTIFY DEVICE and IDENTIFY PACKET DEVICE in ATA8-ACS).

The content of the arguments is defined in 5.1.3.

NOTE 4 - The CFA TRANSLATE SECTOR command is an example where LBA and Count contain command specific information for this protocol (see ATA8-ACS for a description of the CFA TRANSLATE SECTOR command).

5.3.4 PIO Data-Out Command protocol

5.3.4.1 PIO Data-Out Command protocol overview

Processing a **PIO Data-Out Command** procedure call includes the delivery of one or more DRQ data blocks (see 3.1.15) of write data. The **PIO Data-Out Command** protocol is differentiated from the **DMA Command** protocol that transfers write data by having the device make the status available after delivering each DRQ data block, as opposed to having the device make status available after all data for the command is delivered. See the ATA8-ACS standard for description of the PIO data-out commands.

5.3.4.2 PIO Data-Out Command protocol description

Figure 17 shows the processing of the PIO data-out command procedure call. The blue dashed lines and adjacent blue text and numbers in figure 17 show conditional or optional behavior as described in his subclause.

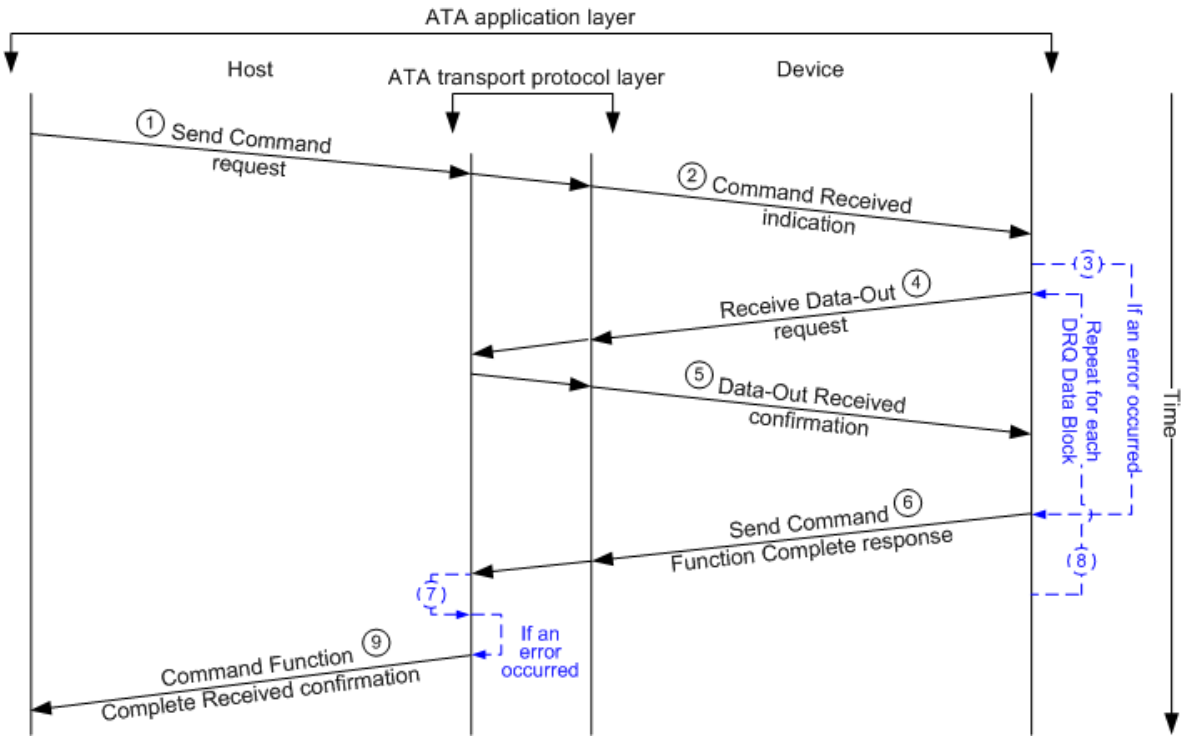

Figure 17 — PIO Data-Out Command procedure call

The following is the description of the sequence of the **PIO Data-Out Command** procedure call:

1) The application client sends a **Send Command (Command, [Device], LBA, Count, [Features], Host Buffer)** request to the host port causing the host port to deliver the request via the service delivery subsystem;
2) After receiving the request, the device port sends a **Command Received (Command, LBA, Count, [Features])** indication to the device server causing the device server to process the command;
3) If an error occurred, then go to step 6;
4) The device server sends a **Receive Data-Out (Count, Device Buffer, DRQ Data Block)** request to the device port causing the device port to receive the write data via the service delivery subsystem. The **Receive Data-Out** request may also contain transport specific information that is transmitted by the device port to the host port;
5) After all of the data has been received for the request, the device port sends a **Data-Out Received ([LBA], Status, [Error])** confirmation to the device server indicating that the data was received or that an error occurred;
6) After receiving the confirmation, the device server sends a **Send Command Function Complete ([LBA], [Interrupt], Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem;
7) If the Error argument was present in the response, and the host port is not delivering data via the service delivery subsystem, then go to step 9. If the Error argument was present in the response, and the host port is delivering data, then the host port may stop delivering data and go to step 9;
8) If there is more data to be received for the command, then go to step 4; and
9) After all of the data has been received for the command or an error occurred, the host port sends a **Command Function Complete Received ([Device], [LBA], Status, [Error])** confirmation to the application client.

The content of the arguments is defined in 5.1.3.

NOTE 5 - The DEVICE CONFIGURATION SET command is an example where LBA and Count contain command specific information for this protocol (see ATA8-ACS for description of the DEVICE CONFIGURATION SET command).

5.3.5 DMA command protocol

5.3.5.1 DMA Command protocol overview

Processing of a **DMA Command** procedure call includes delivery of read data or write data. When used to deliver read data, the **DMA Command** protocol is differentiated from the **PIO Data-In Command** protocol by having the device make the status available after all data is delivered or when an error occurs, as opposed to having the device make status available before delivering each DRQ data block. When used to deliver write data, the **DMA command** protocol is differentiated from the **PIO Data-Out Command** protocol by having the device make the status available after all data is delivered or when an error occurs, as opposed to having the device make the status available after delivering each DRQ data block. See the ATA8-ACS standard for description of the DMA commands.

5.3.5.2 DMA Command protocol description

Figure 18 shows the processing of the **DMA Command** procedure call. The blue dashed lines and adjacent blue text and numbers in figure 18 show conditional or optional behavior as described in this subclause.

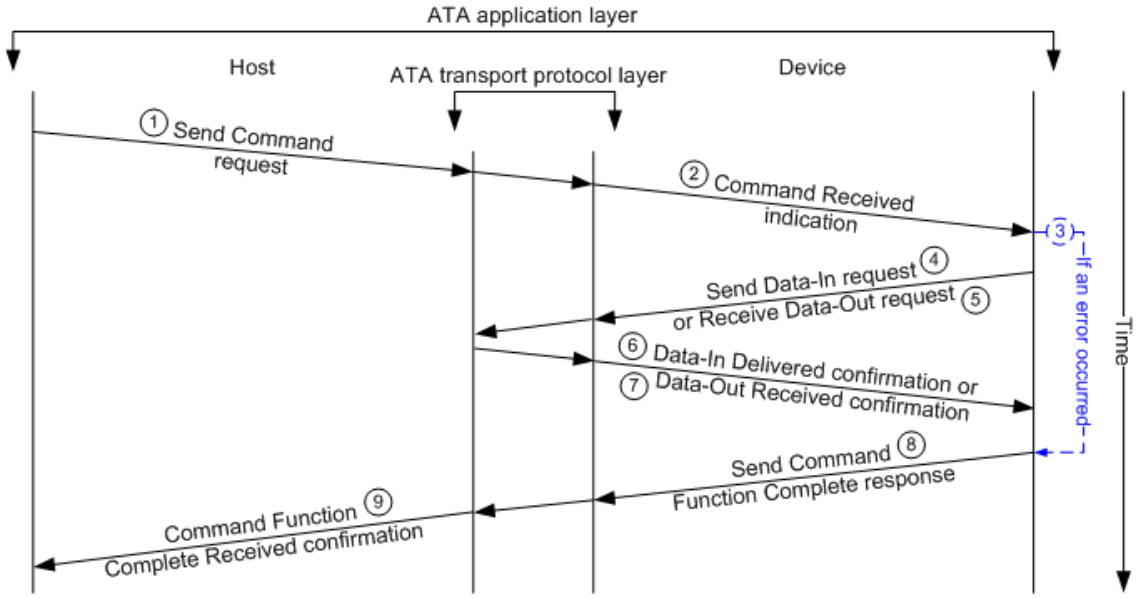

Figure 18 — DMA Command procedure call

The following is the description of the sequence of the **DMA Command** procedure call:

1) The application client sends a **Send Command (Command, [Device], LBA, Count, [Features], Host Buffer)** request to the host port causing the host port to make the request available via the service delivery subsystem;
2) After receiving the request, the device port sends a **Command Received (Command, LBA, Count, [Features], [Host Buffer])** indication to the device server causing the device server to process the command;
3) If an error occurred, then go to step 8.
4) If the command specified that read data is to be transferred, then the device server prepares the data and sends a **Send Data-In (Count, [Host Buffer], Device Buffer)** request to the device port causing the device port to make the data available via the service delivery subsystem. The **Send Data-In**

request may also contain transport specific information that is transmitted by the device port to the host port;

5) If the command specified that write data is to be transferred, then the device server sends a **Receive Data-Out (Count, Device Buffer)** request to the device port causing the device port to receive the data via the service delivery subsystem. The **Receive Data-Out** request may also contain transport specific information that is transmitted by the device port to the host port;
6) If the command specified that read data is to be transferred, then, after all of the data has been delivered for the command, the device port sends a **Data-In Delivered ([LBA], Status, [Error])** confirmation to the device server indicating that the data was delivered or that an error occurred;
7) If the command specified that write data is to be transferred, then, after all of the data has been received for the command, the device port sends a **Data-Out Received ([LBA], Status, [Error])** confirmation to the device server indicating that the data was received or that an error occurred;
8) After receiving the confirmation, the device server sends a **Send Command Function Complete ([LBA], [Interrupt], Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem; and
9) After receiving the response, the host port sends a **Command Function Complete Received ([Device], [LBA], Status, [Error])** confirmation to the application client.

The content of the arguments is defined in 5.1.3.

5.3.6 DMA Queued Command protocol

5.3.6.1 DMA Queued Command protocol overview

The **DMA Queued Command** protocol is differentiated from the **DMA Command** protocol by the ability for the device, after receiving one command from the Overlapped feature set (see ATA8-ACS), to signal the host that the device is not ready to transfer data for a command in that feature set but is able to receive additional commands from that feature set. Processing of a **DMA Queued Command** procedure call may include delivery of read data and/or write data.

5.3.6.2 DMA Queued Command protocol description

Figure 19 shows the processing of the **DMA Queued Command** procedure call. The blue dashed lines and adjacent blue text and numbers in figure 19 show conditional or optional behavior as described in this subclause.

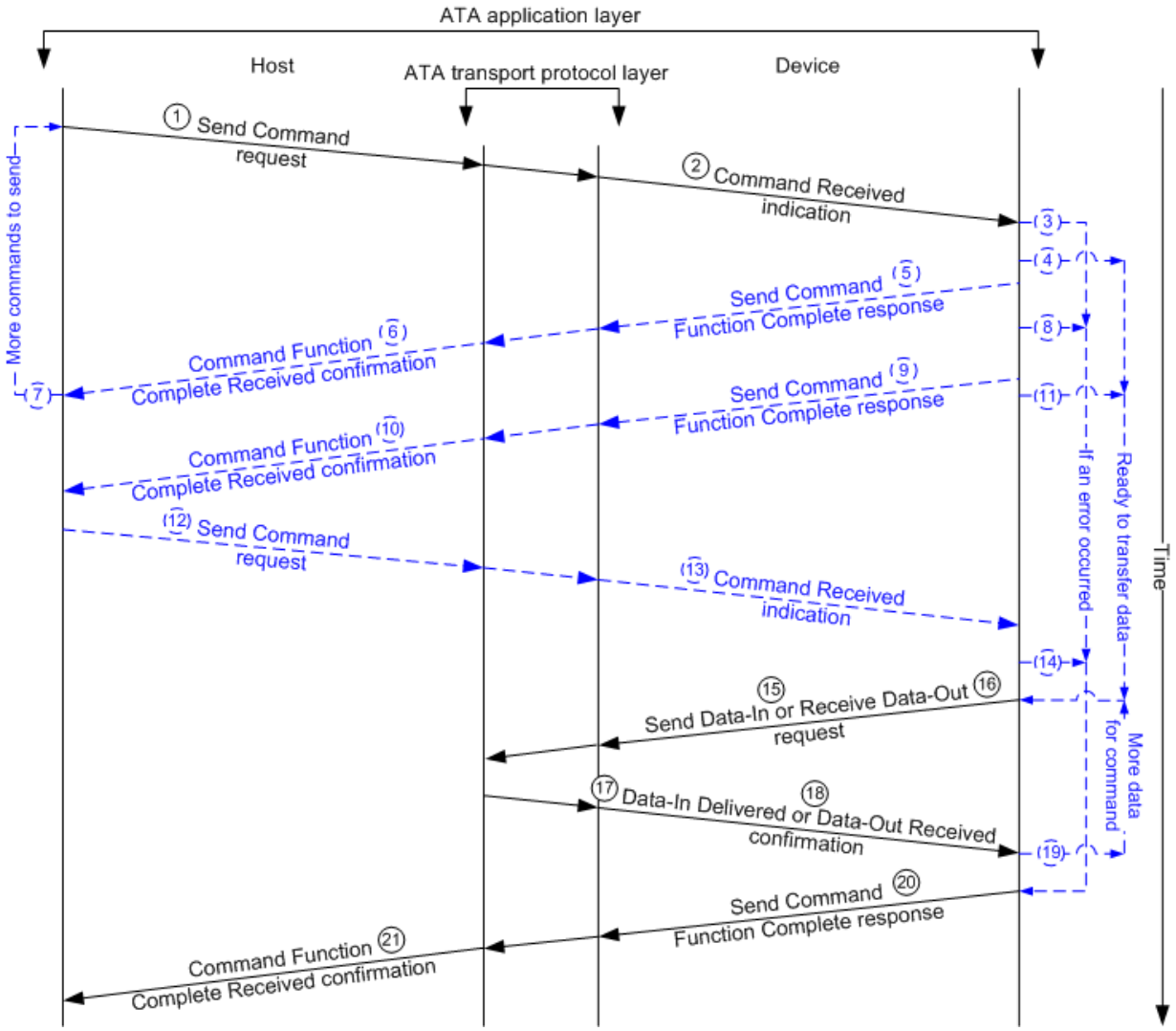

Figure 19 — DMA Queued Command procedure call

The following is the description of the sequence of the **DMA Queued Command** procedure call:

1) The application client sends a **Send Command (Command, [Device], LBA, Count, [Features], Tag, [Host Buffer])** request to the host port causing the host port to make the request available via the service delivery subsystem;
2) After receiving the request, the device port sends a **Command Received (Command, LBA, Count, [Features], Tag, [Host Buffer])** indication to the device server causing the device server to process the command;
3) If an error occurred, then go to step 20;
4) If the device server is ready to transfer data for a queued command, then go to step 15;
5) If the device server is not ready to transfer data for a queued command, then the device server sends a **Send Command Function Complete ([Interrupt], Status)** response to the device port causing the device port to make the response available via the service delivery subsystem;

6) If the host port receives a response via the service delivery subsystem because the device server is not ready to transfer data for a queued command, then the host port sends a **Command Function Complete Received ([Device], Status)** confirmation to the application client;
7) After receiving the confirmation, the application client may either wait to receive a **Command Function Complete Received** confirmation indicating that the device is ready to transfer data for a command, or, if the application client has not sent the maximum number of queued commands allowed by the device, the application client may go to step 1 in order to send another **Send Command** request;
8) If the device server sent a **Send Command Function Complete ([Interrupt], Status)** response to the device port because the device server was not ready to transfer data for a queued command, and an error occurred for a queued command for which data has not been delivered, then go to step 20;
9) If the device server sent a **Send Command Function Complete ([Interrupt], Status)** response to the device port because the device server was not ready to transfer data for a queued command, and the device server is now ready to transfer data for a queued command, then the device server sends a **Send Command Function Complete ([Interrupt], Tag, Status)** response to the device port causing the device port to make the response available via the service delivery subsystem;
10) If the host port receives a response, then the host port sends a **Command Function Complete Received ([Device], Tag, Status)** confirmation to the application client;
11) If a command is not required to initiate data delivery, then go to step 15;
12) If a command is required to initiate data delivery, and the application client is ready to transfer data for a queued command for which the application client has received a **Command Function Complete Received** confirmation indicating that the device port is ready for the data transfer, then the application client sends a **Send Command (Command, [Device])** request to the host port causing the host port to make the request available via the service delivery subsystem (the Command argument in this step contains the code for the SERVICE command (see ATA8-ACS));
13) After receiving the request, the device port sends a **Command Received (Command)** indication to the device server causing the device server to process the command;
14) If an error occurred, then go to step 20;
15) If the command specified that read data is to be transferred, then the device server prepares the data and sends a **Send Data-In (Count, Tag, [Host Buffer], Device Buffer)** request to the device port causing the device port to make the data available via the service delivery subsystem;
16) If the command specified that write data is to be transferred, then the device server sends a **Receive Data-Out (Count, Tag, Device Buffer)** request to the device port causing the device port to receive the data via the service delivery subsystem;
17) If the command specified that read data is to be transferred, then, after all of the data has been delivered for the command, the device port sends a **Data-In Delivered ([LBA], Tag, Status, [Error])** confirmation to the device server indicating that the data was delivered or that an error occurred;
18) If the command specified that write data is to be transferred, then, after all of the data has been received, the device port sends a **Data-Out Received ([LBA], Tag, Status, [Error])** confirmation to the device server indicating that the data was received or that an error occurred;
19) If additional data transfers are required for the command, then go to step 15;
20) After receiving the confirmation, the device server sends a **Send Command Function Complete ([LBA], [Interrupt], Tag, Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem; and
21) After receiving the response, the host port sends a **Command Function Complete Received ([Device], [LBA], Tag, Status, [Error])** confirmation to the application client.

The content of the arguments is defined in 5.1.3.

5.3.7 PACKET Command protocol

5.3.7.1 PACKET Command protocol overview

The **PACKET Command** protocol allows a host to send a command to a device via a command packet. The command packet contains the command and command parameters that the device is to process (see the ATA8-ACS standard for description of the PACKET command). After receiving the command, the device server

parses the command and processes the command depending on whether an error occurred or the actions required by the command.

5.3.7.2 PACKET Command protocol description

5.3.7.2.1 PACKET command transfer

Figure 20 shows the transfer of a PACKET command.

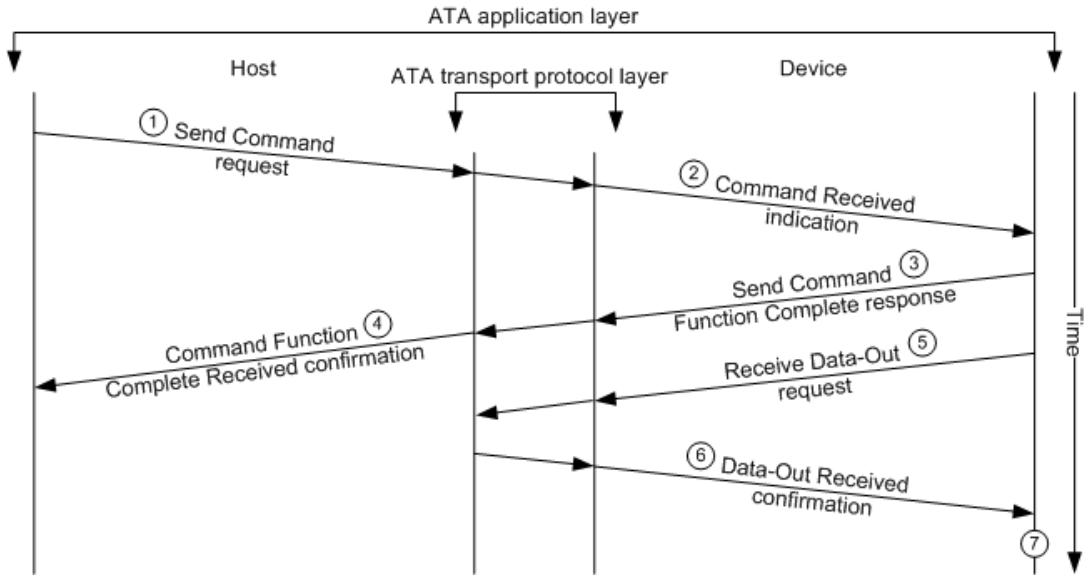

Figure 20 — PACKET command transfer

The following is the description of the sequence of the PACKET command transfer:

1) The application client sends a **Send Command (Command, [Device], Count, [Features], Host Buffer)** request to the host port causing the host port to make the request available via the service delivery subsystem;
2) After receiving the request, the device port sends a **Command Received (Command, Count, [Features])** indication to the device server causing the device server to process the command;
3) The device server sends a **Send Command Function Complete (Count, [Interrupt], Input/Output, Command/Data, Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem;
4) After receiving the response, the host port sends a **Command Function Complete Received ([Device], Count, Status, [Error])** confirmation to the application client;
5) If no error occurred, then the device server sends a **Receive Data-Out (Count, Device Buffer)** request to the device port causing the device port to receive the write data via the service delivery subsystem;
6) After all of the data has been received for the request, the device port sends a **Data-Out Received (Status, [Error])** confirmation to the device server indicating that data was received or that an error occurred;
7) The device server parses the command received during the data delivery;
8) If an error occurred or the command requires no data transfer, then go to 5.3.7.2.2;
9) If the command requires transfer of read data using PIO, then go to 5.3.7.2.3;
10) If the command requires transfer of write data using PIO, then go to 5.3.7.2.4; and
11) If the command requires data transfer using DMA, then go to 5.3.7.2.5.

The content of the arguments is defined in 5.1.3.

5.3.7.2.2 PACKET command completion with no data transfer

Figure 21 shows completion of a PACKET command with no data transfer.

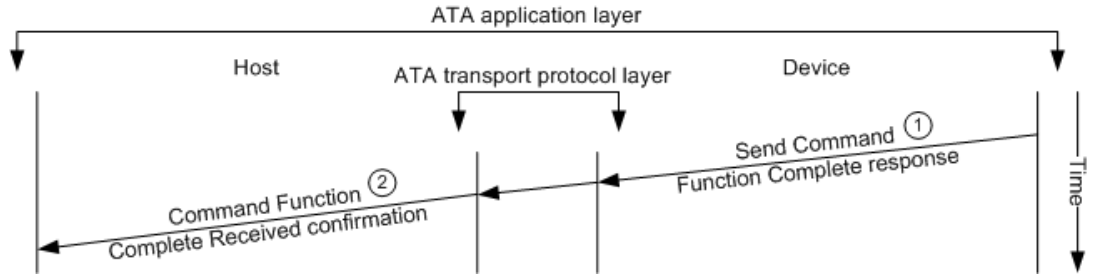

Figure 21 — PACKET command completion with no data transfer

The following is the description of the sequence of the completion of a PACKET command with no data transfer:

1) The device server sends a **Send Command Function Complete (Input/Output, Command/Data, Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem; and
2) After receiving the response, the host port sends a **Command Function Complete Received ([Device], Status, [Error])** confirmation to the application client.

The content of the arguments is defined in 5.1.3.

5.3.7.2.3 PACKET command completion with read data transfer using PIO

Figure 22 shows completion of a PACKET command with transfer of read data using PIO. The blue dashed lines and adjacent blue text and numbers in figure 22 show conditional or optional behavior as described in this subclause.

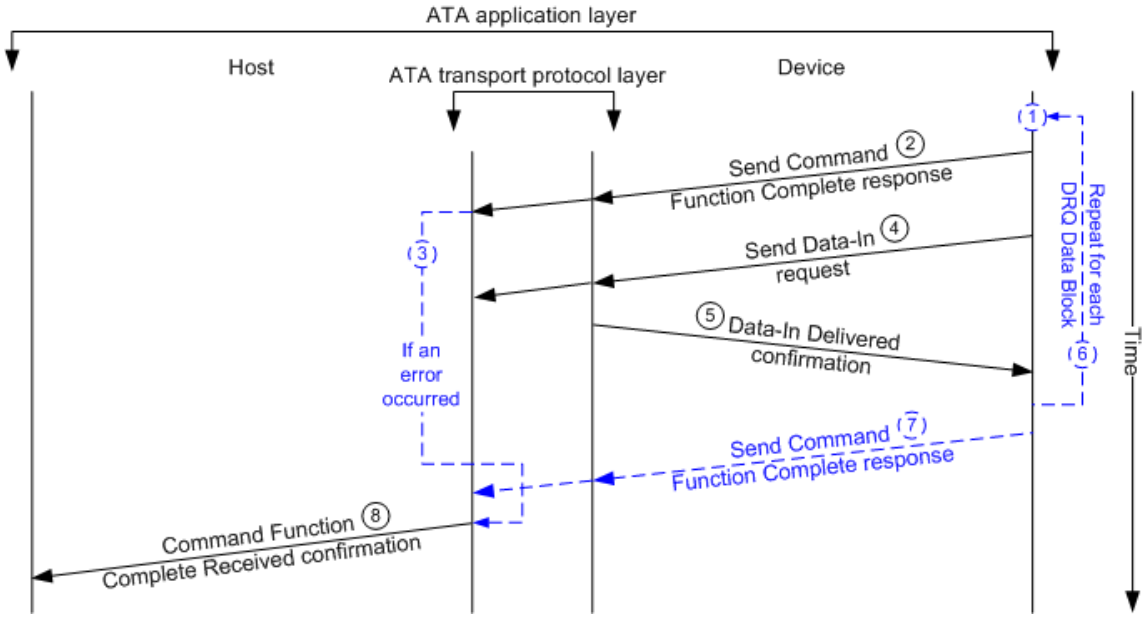

Figure 22 — PACKET command completion with read data transfer using PIO

The following is the description of the sequence of the completion of a PACKET command with transfer of read data using PIO:

1) The device server prepares the read data for the command;

2) The device server sends a **Send Command Function Complete (Count, [Interrupt], Input/Output, Command/Data, Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem;
3) If the Error argument was present in the response, then the device server may stop data delivery and go to step 7;
4) The device server sends a **Send Data-In (Count, [Host Buffer], Device Buffer, DRQ Data Block)** request to the device port causing the device port to make the read data available via the service delivery subsystem. If this is not the first DRQ data block for the command and Host Buffer is required for this service, then Host Buffer contains the value from the previous Host Buffer plus the number of bytes delivered in the previous DRQ data block. The **Send Data-In** request may also contain transport specific information that is transmitted by the device port to the host port;
5) After all of the data has been delivered for the request, the device port sends a **Data-In Delivered (Status, [Error])** confirmation to the device server indicating that data was delivered or that an error occurred;
6) If there is more data to be delivered for the command, then go to step 1;
7) The device server may send a transport specific **Send Command Function Complete (Count, [Interrupt], Input/Output, Command/Data, Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem; and
8) After all of the data has been delivered for the command or an error occurred, the host port sends a **Command Function Complete Received ([Device], Status, [Error])** confirmation to the application client.

The content of the arguments is defined in 5.1.3.

5.3.7.2.4 PACKET command completion with write data transfer using PIO

Figure 23 shows completion of a PACKET command with transfer of write data using PIO. The blue dashed lines and adjacent blue text and numbers in figure 23 show conditional or optional behavior as described in this subclause.

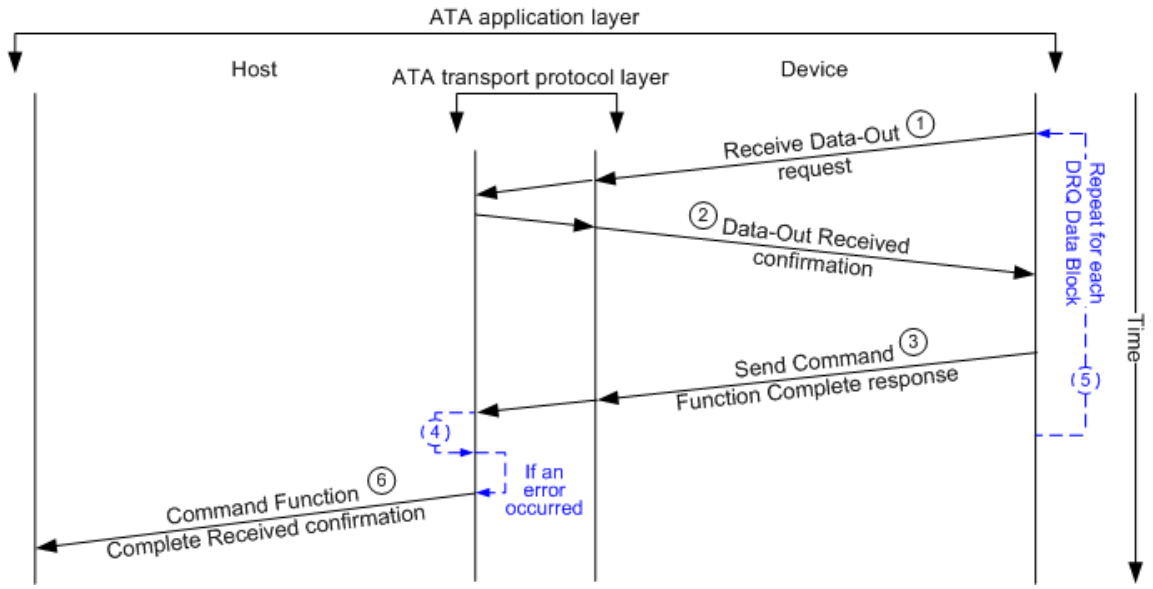

Figure 23 — PACKET command completion with write data transfer using PIO

The following is the description of the sequence of the completion of a PACKET command with transfer of write data using PIO:

1) The device server sends a **Receive Data-Out (Count, Device Buffer, DRQ Data Block)** request to the device port causing the device port to receive the data via the service delivery subsystem. The

Receive Data-Out request may also contain transport specific information that is transmitted by the device port to the host port;

2) After all of the data has been received for the request, the device port sends a **Data-Out Received (Status, [Error])** confirmation to the device server indicating that the data was received or that an error occurred;
3) The device server sends a **Send Command Function Complete (Count, [Interrupt], Input/Output, Command/Data, Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem;
4) If the Error argument was present in the response, and the host port is not delivering data via the service delivery subsystem, then go to step 6. If the Error argument was present in the response, and the host port is delivering data, then the host port may stop delivering data and go to step 6;
5) If there is more data to be received for the command, then go to step 1; and
6) After all of the data has been received for the command or an error occurred, the host port sends a **Command Function Complete Received ([Device], Status, [Error])** confirmation to the application client.

The content of the arguments is defined in 5.1.3.

5.3.7.2.5 PACKET command completion with data transfer using DMA

Figure 24 shows completion of a PACKET command with data transfer using DMA.

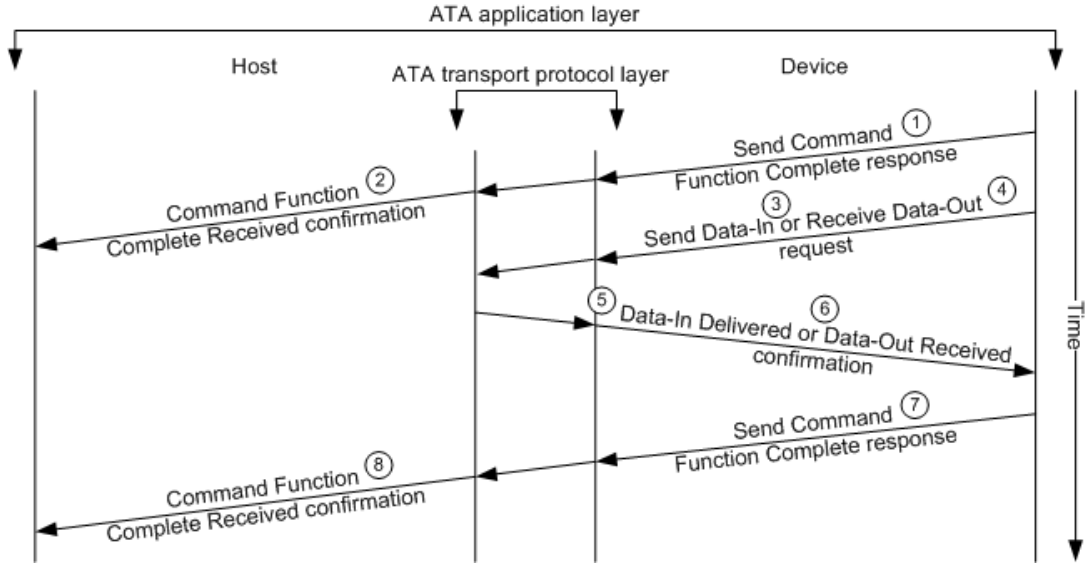

Figure 24 — PACKET command completion with data transfer using DMA

The following is the description of the sequence of the completion of a PACKET command with transfer of data using DMA:

1) The device server sends a **Send Command Function Complete (Count, [Interrupt], Input/Output, Command/Data, Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem;
2) After receiving the response, the host port sends a **Command Function Complete Received ([Device], Status, [Error])** confirmation to the application client;
3) If the command specified that read data is to be transferred, then the device server prepares the data and sends a **Send Data-In (Count, [Host Buffer], Device Buffer)** request to the device port causing the device port to make the data available via the service delivery subsystem. The **Send Data-In** request may also contain transport specific information that is transmitted by the device port to the host port;
4) If the command specified that write data is to be transferred, then the device server sends a **Receive Data-Out (Count, Device Buffer)** request to the device port causing the device port to receive the

data via the service delivery subsystem. The **Receive Data-Out** request may also contain transport specific information that is transmitted by the device port to the host port;

5) If the command specified that read data is to be transferred, then, after all the data has been delivered for the command, the device port sends a **Data-In Delivered (Status, [Error])** confirmation to the device server indicating that the data was delivered or that an error occurred;
6) If the command specified that write data is to be transferred, then, after all of the data has been received for the command, the device port sends a **Data-Out Received (Status, [Error])** confirmation to the device server indicating that the data was received or that an error occurred;
7) After receiving the confirmation, the device server sends a **Send Command Function Complete ([Interrupt], Input/Output, Command/Data, Status, [Error])** response to the device port causing the device port to make the response available via the service delivery subsystem; and
8) After receiving the response, the host port sends a **Command Function Complete Received ([Device], Status, [Error])** confirmation to the application client.

The content of the arguments is defined in 5.1.3.