Working Draft
American National Standard

Project
T13/1700-D

Revision 2
04 November 2005

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

T13 Technical Editor:

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

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

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

Points of Contact

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

T13 Chair

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

Telephone: 702-614-6119

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 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.

Contents

Tables

Page

Figures

Page

Foreword

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

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

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

Clause 4 describes the overall ATA architectural model.

Clause 5 describes the ATA protocol model for the ATA architecture.

American National Standard –
For Information Technology

AT Attachment – 8 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 interoperability.

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 5.3).

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.

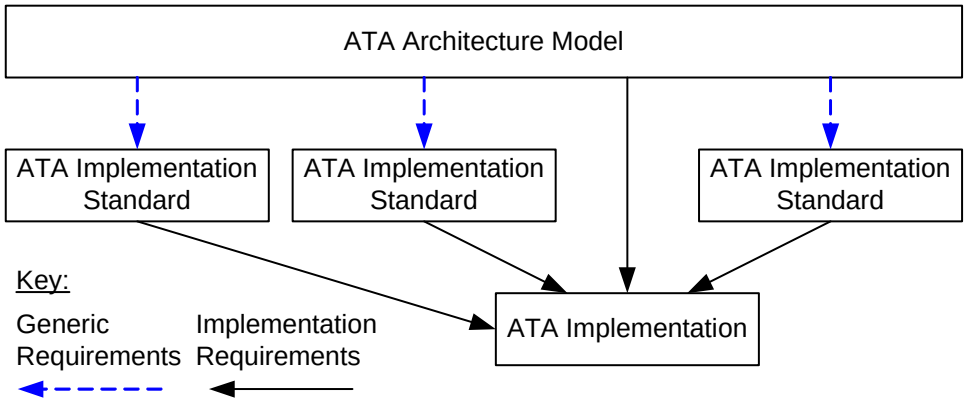

Figure 1 — Requirements precedence

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 ATA compliance shall conform to the applicable implementation requirements defined in this standard and the appropriate ATA implementation standards. In the event of a conflict between this document and other ATA standards under the jurisdiction of technical committee T13, the requirements of this standard shall apply.

1.3 ATA family of standards

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

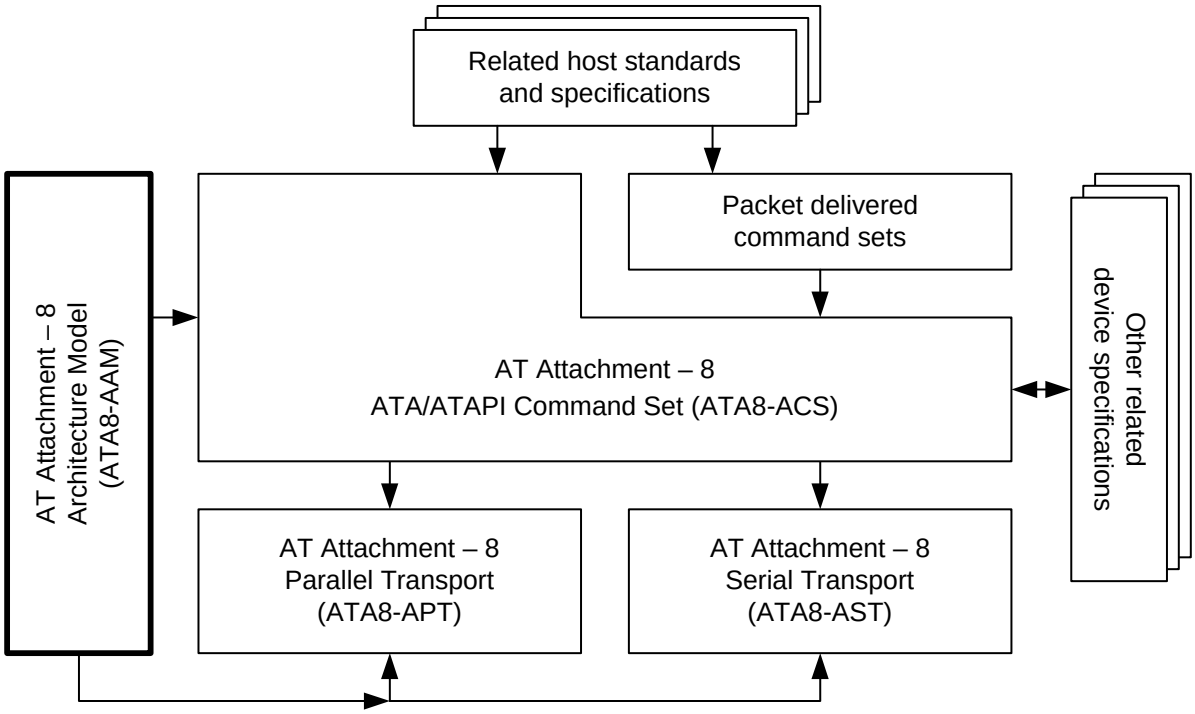

Figure 2 — Standards and specifications relationships

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

Table 1 — Related host standards and specifications examples

Protected Area Run Time Interface Extension Services (PARTIES) standard [ANSI NCITS 346-2001]
ATA/ATAPI Host Adapter Standards (ATA Adapter) standard [ANSI NCITS 370-2004]
BIOS Enhanced Disk Drive Services – 3 (EDD-3) draft standard [T13/1572-D]

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.

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

Table 2 — Packet delivered command set examples

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

Other related device specifications: define other implementations that use elements of the ATA standards. Table 2 shows examples of other related device specifications.

Table 3 — Other related device specifications

Serial ATA: High Speed Serialized AT Attachment, Revision 1.0a
CF+ and CompactFlash™ Specification, Revision 3.0
MultiMediaCard MCA System Specification, Version 4.1
CE-ATA Digital Protocol, Revision 1.0

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.

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.

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 the following documents may be obtained from ANSI:

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

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

Table 4 shows standards bodies and their web sites.

Table 4 — Standards bodies

Abbreviation	Standards body	Web site
ANSI	American National Standards Institute	http://www.ansi.org
IEC	International Engineering Consortium	http://www.iec.ch
IEEE	Institute of Electrical and Electronics Engineers	http://www.ieee.org
INCITS	International Committee for Information Technology Standards	http://www.incits.org
ISO	International Standards Organization	http://www.iso.ch
ITI	Information Technology Industry Council	http://www.itic.org
T10	INCITS T10 SCSI storage interfaces	http://www.t10.org
T11	INCITS T11 Fibre Channel interfaces	http://www.t11.org
T13	INCITS T13 ATA storage interface	http://www.t13.org

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 (part 1 of 2)

AT Attachment with Packet Interface Extension (ATA/ATAPI-4) [ANSI INCITS 317-1998]
Protected Area Run Time Interface Extension Services (PARTIES) standard [ANSI NCITS 346-2001]
ATA/ATAPI Host Adapter Standards (ATA Adapter) standard [ANSI NCITS 370-2004]
ISO/IEC 14776-364, SCSI Multimedia Commands – 4 (MMC-4) standard [ANSI/INCITS 401-2005]
ISO/IEC 14776-413, SCSI Architecture Model – 3 (SAM-3) standard [ANSI/INCITS 402-2005

Table 5 — Approved normative references (part 2 of 2)

ISO/IEC 14776-322, SCSI Block Commands – 2 (SBC-2) standard [ANSI/INCITS 405-2005]
BIOS Enhanced Disk Drive Services – 3 (EDD-3) standard [ANSI/INCITS 407-2005]

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

ISO/IEC 14776-453, SCSI Primary Commands – 3 (SPC-3) draft standard [Project T10/1416-D]
ISO/IEC 14776-151, Serial Attached SCSI – 1.1 (SAS-1.1) draft standard [Project T10/1601-D]
SCSI / ATA Translation (SAT) draft standard [Project T10/1711-D]
AT Attachment-8 Serial Transport (ATA8-AST) draft standard [Project T13/1697-D]
AT Attachment-8 Parallel Transport (ATA8-APT) draft standard [Project T13/1698-D]
AT Attachment-8 ATA/ATAPI Command Set (ATA8-ACS) draft standard [Project T13/1699-D]

2.3 Other references

2.3.1 Other references overview

The technical reports and specifications listed in this clause are referenced in this standard.

2.3.2 Technical reports

Table 7 lists the technical reports referenced in this standard. Copies of the published technical reports may be obtained from ANSI (see 2.2.1). At the time of publication, the technical report drafts listed in table 7 were still under development. For information on the current status of the project, or regarding availability, contact the relevant standards body or other organization as indicated.

Table 7 — Technical reports

Address Offset Alternate Boot Feature technical report [ANSI/INCITS TR27-2001]
Time Limited Commands technical report [ANSI/INCITS TR37-2004]
SMART Command Transport (SCT) draft technical report [Project T13/1701-DT-N]

2.3.3 Other specifications

Table 8 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 8 — Other specifications (part 1 of 2)

Specification	Contact information for obtaining specification
Serial ATA: High Speed Serialized AT Attachment Revision 1.0a	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.

Table 8 — Other specifications (part 2 of 2)

Specification	Contact information for obtaining specification
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 1.0	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 a host that is the source of application client tasks. (See 4.2.2.)

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

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

3.1.4 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.)

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

3.1.6 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.)

3.1.7 device: a storage peripheral that processes commands and device management functions (see 4.2.3). A device may either be an ATA device (see 3.1.3) or an ATAPI device (see 3.1.5).

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

3.1.9 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.)

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

3.1.11 domain: an ATA domain (see 3.1.4).

3.1.12 DRQ data block: a unit of data words associated with available status when using either the PIO data-in command protocol or PIO data-out command protocol.

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

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

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

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

3.1.17 nexus loss event: a transport specific event where a 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.)

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

3.1.19 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. (See 4.2.4.)

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

3.2 Symbols

3.3 Abbreviations

I/O	Input/Output
LBA	Logical Block Address (see 3.1.16)

3.4 Keywords

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

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

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

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

3.4.5 may: a keyword that indicates flexibility of choice with no implied preference (as distinguished from the definition for “should” (see 3.4.13)).

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

3.4.7 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.

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

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

3.4.10 reserved for: a keyword used to describe bits, bytes, words, fields, and code values that are set aside for use in other standards or for other data structures in this standard.

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

3.4.12 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.

3.4.13 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 3.4.5)).

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

3.5 Conventions

3.5.1 Editorial conventions

Lowercase is used for words having the normal English meaning. Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 3 or in the text where they first appear.

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

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

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

3.5.2 Numeric conventions

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

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

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

This standard uses the American convention for representing decimal numbers (e.g., the thousands and higher multiples are separated by a comma, and a decimal point is used to separate the ones and higher digits and the tenths and smaller digits). Table 9 shows some examples of decimal numbers using the American and International Standards Organization (ISO) numbering conventions.

Table 9 — American and ISO numbering conventions

American	ISO
0.6	0,6
3.14159265	3,141 592 65
1,000	1 000
1,323,462.95	1 323 462,95

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

3.5.3 List notation

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

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

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

1) First occurrence;
   A) Item A for first occurrence; or (or and)
   B) Item B for first occurrence;
2) Second occurrence; and
3) Third occurrence.

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

1) First occurrence;
   i) First nested occurrence for first occurrence; and
   ii) Second nested occurrence for first occurrence;
2) Second occurrence; and
3) Third occurrence.

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

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

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

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

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

a) Item a:
   i) First occurrence for item a; and
   ii) Second occurrence for item a;
b) Item b; or (or and)
c) Item c.

3.5.4 Precedence

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

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

4 ATA architecture model

4.1 Introduction

The purpose of the ATA architecture model is to:

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

The development of this standard is assisted by the use of an abstract model. To 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 the device with a command code value). The Command object is instantiated in the serial ATA transport protocol by the command code value placed in the Command Frame Information Structure (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. 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 powered on and capable of communicating with each

other through the service delivery subsystem. Figure 3 shows a schematic of an example of a simple ATA domain.

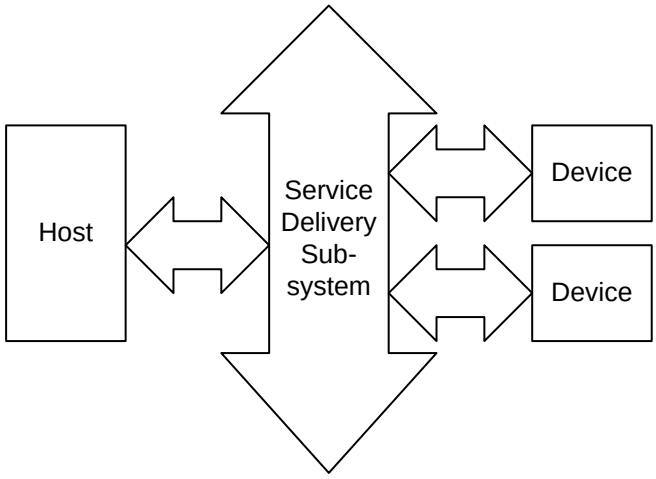

Figure 3 — Simple ATA domain example

NOTE 1 - there are other objects that may be in an ATA domain (e.g. SATA port selectors, SATA port multipliers, and Serial ATA 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 the 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, and data to devices; and
c) receives data and status from devices.

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

A host contains application clients and a host port. An application client is an object in a host that is the source of commands and device management requests. An application client is independent of the interconnect and ATA transport protocol (e.g., an application client may correspond to the device driver and any other code within the operating system that is capable of managing I/O requests without requiring knowledge of the interconnect or ATA transport protocol). An application client creates one or more application client tasks, each of which processes a single command or device management function. Application client tasks are part of their parent application client. The application client task directs requests to a remote server (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.

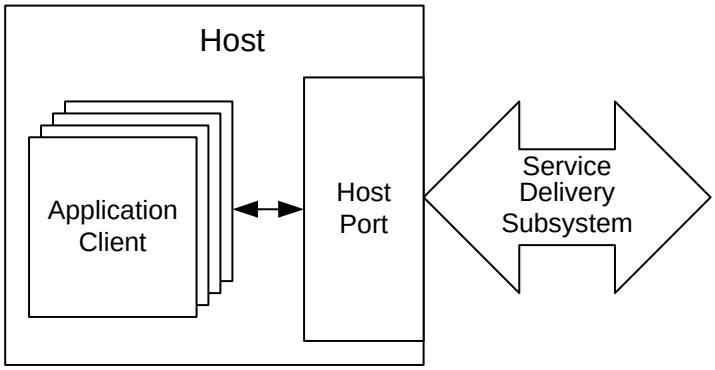

Figure 4 — Simple host schematic

4.2.3 The device

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 processes commands and device management functions, receives data from a host, and transmits data and status to a host.

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.

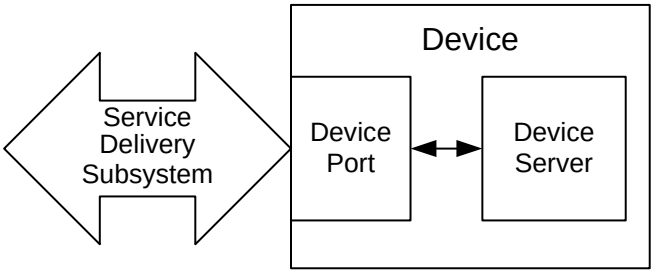

Figure 5 — Simple device schematic

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.

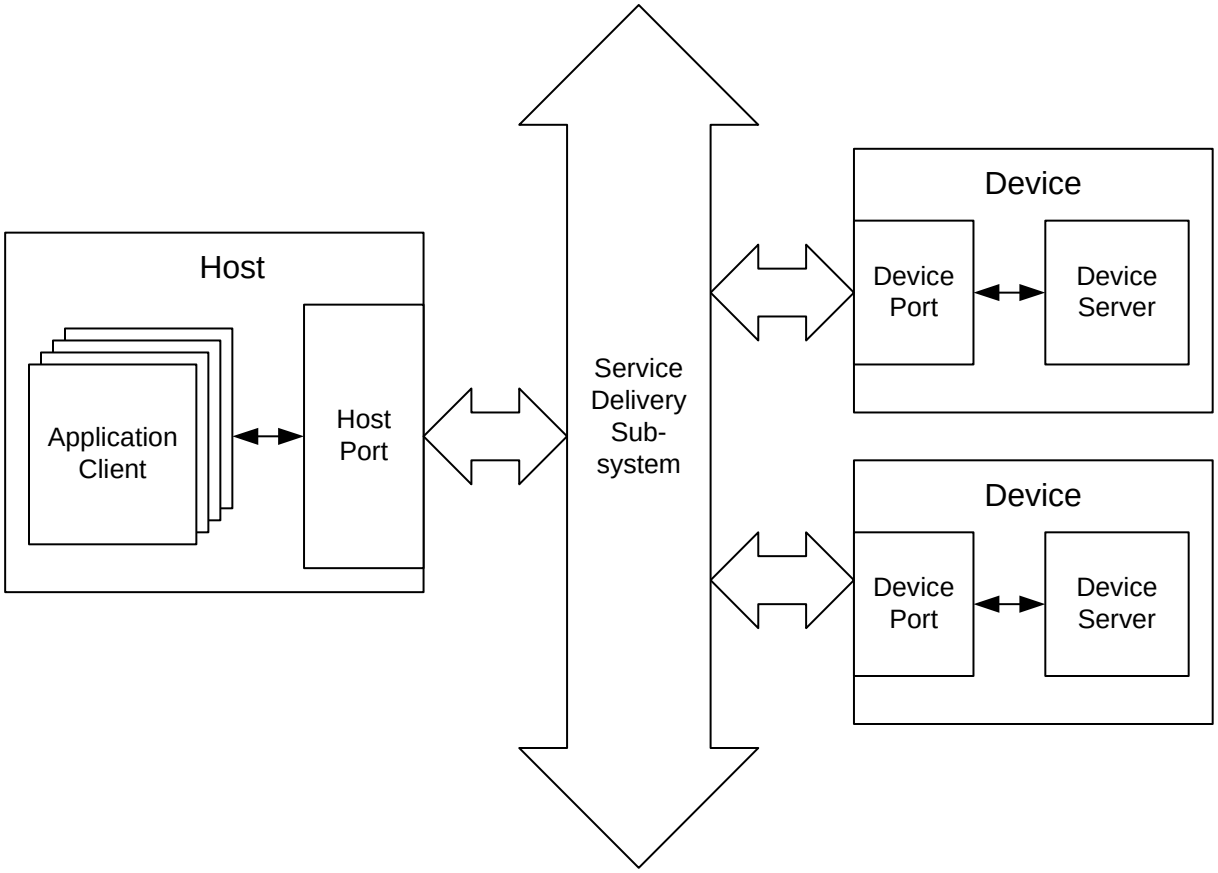

Figure 6 — Example ATA domain with two devices

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.

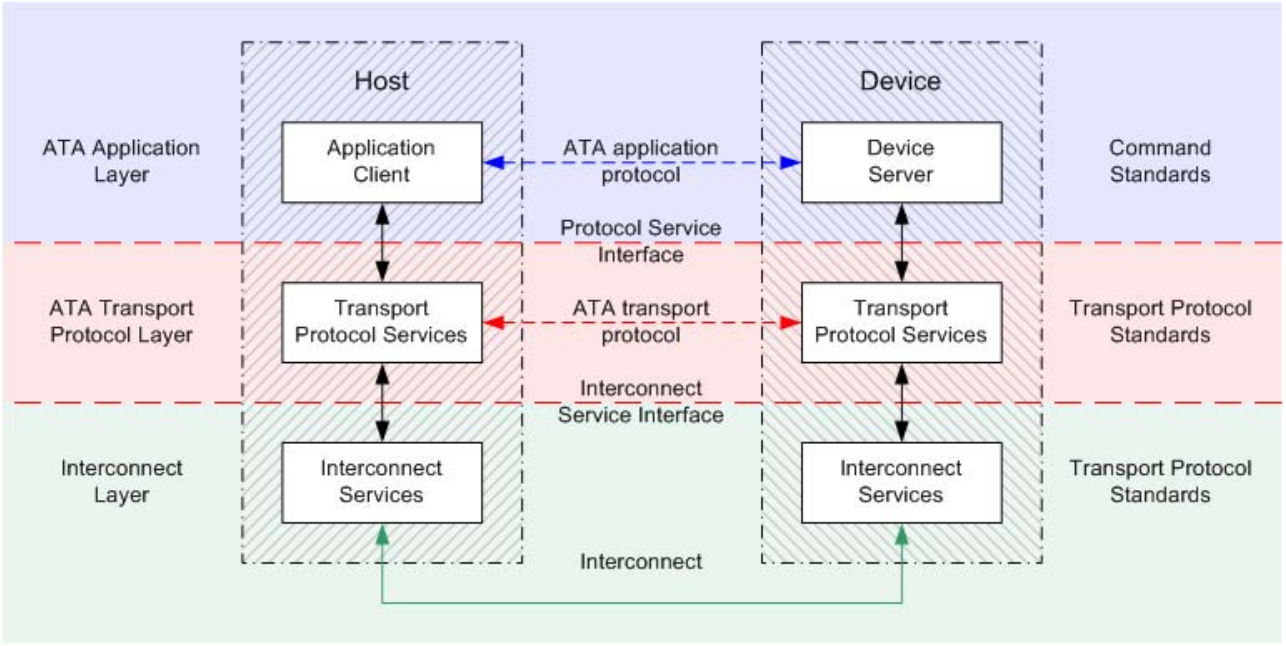

Figure 7 — ATA architecture layering model

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 devices 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 clients that originate and the device servers that process ATA I/O operations 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.

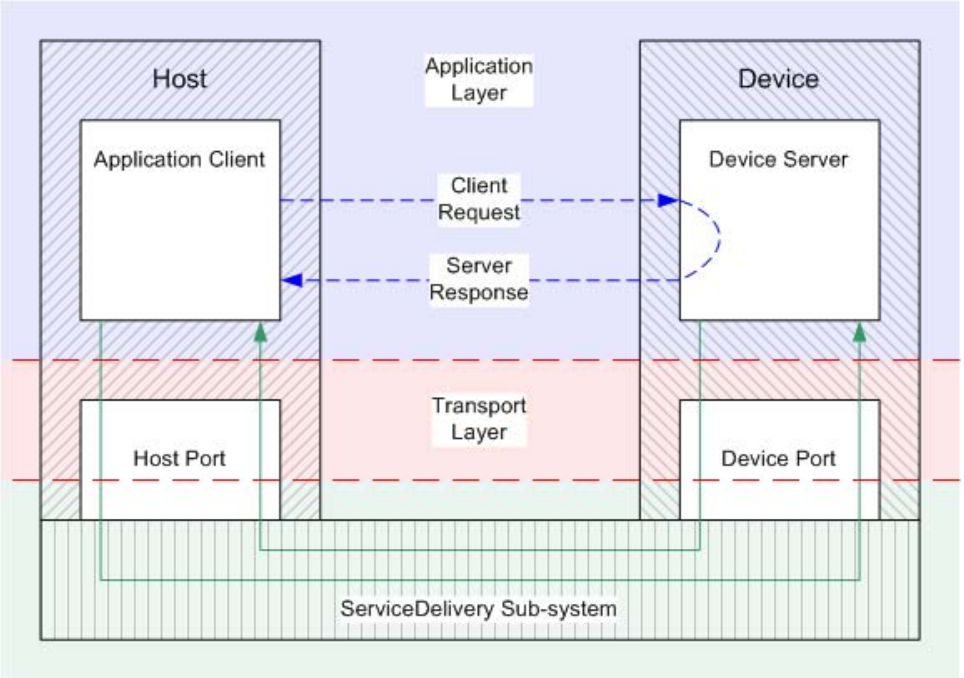

Figure 8 — Client-server model

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 5). All requests originate from the application client residing within a host. An application client creates an application client task to process a request. Each request takes the form of a procedure call with arguments. 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 request becomes pending when it is passed to the host port for transmission. The request is complete when the server response is received. As seen by the device server, the request becomes pending upon receipt and completes when the response is passed to the device port for return to the application client. As a result there may be a time skew between the application client's and the device server's perception of request status and server 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 request or response is:

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

Client-server relationships are not symmetrical. An application client may only originate requests for service. A device server may only respond to such requests. The application client requests an operation provided by a

device server located in a device and waits for completion, which includes transmission of the request to and response from the remote server. In this model, confirmation of successful request or response delivery by the sender is not required. The model assumes that delivery failures are detected by the host port or within the service delivery subsystem.

4.5 The ATA distributed service model

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).

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.

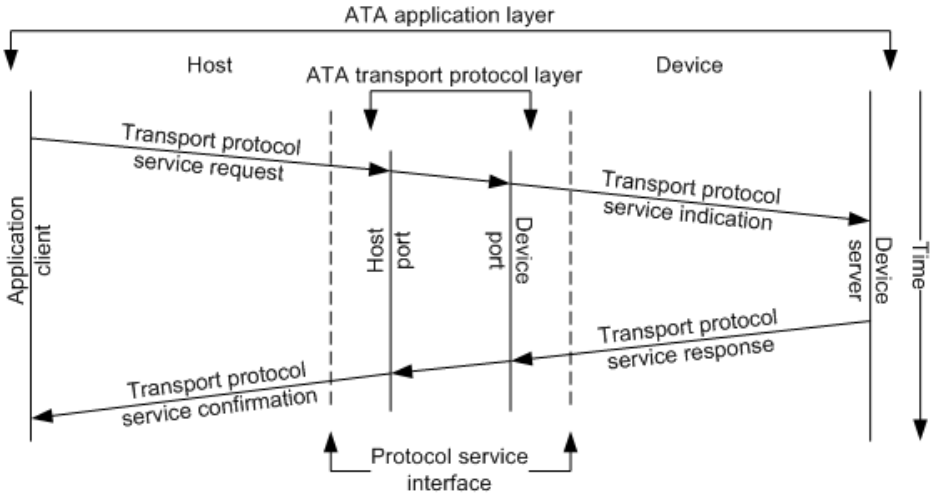

Figure 9 — Transport protocol services

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.

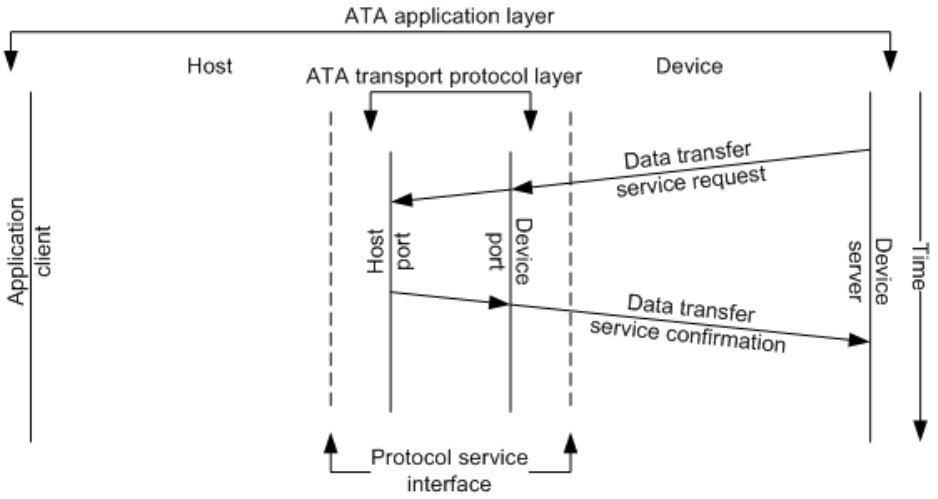

Figure 10 — Data transfer protocol services

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 protocols, including:
   A) DMA data-in command protocol (see 5.3.5);
   B) DMA data-out command protocol (see 5.3.6);
e) DMA queued command protocol (see 5.3.7);
f) Packet command protocols (see 5.3.8); including
   A) Packet non-data command protocol (see 5.3.8.2);
   B) Packet PIO data-in command protocol (see 5.3.8.3);
   C) Packet PIO data-out command protocol (see 5.3.8.4); and
   D) Packet DMA command protocol (see 5.3.8.5).

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 transport protocol services

Each procedure call invokes a sequence of transport services. Table 10 lists each transport protocol service and data transfer protocol service defining its type, origin, destination, and describing the notification the service provides.

Table 10 — ATA transport protocol services (part 1 of 2)

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.

Table 10 — ATA transport protocol services (part 2 of 2)

Name	Type	Sent from	Sent to	Description of the notification
Send Management Function Request	Transport protocol service request	Application client	Host port	The host port is to send a request for a device management transaction sequence.
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 send a completion response for a device management function.
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 send a request for a command sequence.
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 send a completion response for a command function.
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	Data-in data is ready for transfer. Transport-specific information may be sent to the host port.
Data-In Delivered	Data transfer protocol service confirmation	Device port	Device server	A data-in data transfer is complete.
Receive Data-Out	Data transfer protocol service request	Device server	Device port	The host is to send data-out data. Transport-specific information may be sent to the host port.
Data-Out Received	Data transfer protocol service confirmation	Device port	Device server	Data-out data was successfully received from the host or an error occurred while attempting to receive the data.

5.1.3 ATA protocol service arguments

Each transport protocol service or data transfer protocol service contains one or more arguments. Table 11 lists each argument and its definition.

Table 11 — ATA protocol service arguments

Argument	Description
Power-on/Hard-ware Reset	a) Indicates to an application client that a power-on event occurred; or b) Specifies that a device is to perform its hard reset sequence.
Nexus Loss	Indicates to an application client that a nexus loss event has occurred (see 5.2.2).
Software Reset	Specifies that a device is to perform its software reset sequence.
Enable Interrupts	Specifies that a device is to enable its transport-specific interrupt function.
Disable Interrupts	Specifies that a device 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 transferred if only one logical block is to be transferred; b) Specifies the first LBA containing the data to be transferred in a consecutive sequence of LBAs if more than one logical block is to be transferred; c) Specifies or indicates command-specific information (e.g., for SMART commands, a value that is unique to that feature set); or d) Indicates the first LBA of an error.
Count	a) Specifies the number of logical blocks to be transferred for a command; b) Specifies the number of bytes to be transferred for a Packet command function; or c) Specifies or indicates command-specific information (e.g., the transfer mode specified in a SET FEATURES command).
Features	Specifies or indicates additional command-specific information.
Tag	Specifies or indicates a number assigned to a particular command.
Input/Output	Indicates the direction of data transfer in the Packet command protocols (see 5.3.8).
Command/Data	Indicates whether a transfer is for a command or data in the Packet command protocols (see 5.3.8).
DRQ Data Block	Specifies or indicates the amount of data to be transferred for a PIO command function (see 5.3.3, 5.3.4, 5.3.8.3, and 5.3.8.4).
Host Buffer	Contains the beginning address of the buffer in the host to where read data or from where write data is to be transferred for a command.
Device Buffer	Contains the beginning address of the buffer in the device from where read data or to where write data is to be transferred for a command.
Status	Indicates information about command completion and/or the condition of a device.
Error	Indicates additional information if a function resulted in an error.

Table 12 shows which arguments are used for which transport protocol services for which protocols and, when used for a transport protocol service for a protocol, which arguments are mandatory or optional.

Table 12 — 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	Tag	Input/Output	Command/Data	DRQ Data Block	Host Buffer	Device Buffer	Status	Error
Transport Event Notification Received	PR-M	NL-M	–	–	–	–	NL-O	–	–	–	–	–	–	–	–	–	–	–
Send Management Function Request	–	–	DM-O	DM-O	DM-O	–	DM-O	–	–	–	–	–	–	–	–	–	–	–
Management Function Request Received	PR-M	–	DM-O	DM-O	DM-O	–	–	–	–	–	–	–	–	–	–	–	–	–
Send Management Function Complete	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	PR-M DM-M	PR-O DT-O
Management Function Complete Received	–	–	–	–	–	–	PR-O DM-O	–	–	–	–	–	–	–	–	–	PR-M DM-M	PR-O DT-O
Send Command	–	–	–	–	–	AC-M	AC-O	ND-O DC-M DQ-O	ND-O DC-M DQ-O AP-M	AC-O	DQ-O	–	–	–	DC-O DQ-O	–	–	–
Command Received	–	–	–	–	–	AC-M	–	ND-O DC-M DQ-O	ND-O DC-M DQ-O AP-M	AC-O	DQ-O	–	–	–	DC-O DQ-O	–	–	–
Send Command Function Complete	–	–	–	–	–	–	–	ND-O DC-O DQ-M	AP-M	–	DQ-O	AP-M	AP-M	–	–	–	AC-M	AC-O
Command Function Complete Received	–	–	–	–	–	–	AC-O	ND-O DC-O DQ-M	AP-M	–	DQ-O	–	–	–	–	–	AC-M	AC-O
Send Data-In	–	–	–	–	–	–	–	–	DC-M DQ-M PD-M	–	–	–	–	PD-M	DC-O DQ-O PD-O	DC-M DQ-M PD-M	–	–
Data-In Delivered	–	–	–	–	–	–	–	DC-O DQ-O	–	–	–	–	–	–	–	–	DC-M DQ-M PD-M	DC-O DQ-O PD-O
Receive Data-Out	–	–	–	–	–	–	–	–	DC-M DQ-M PD-M	–	–	–	–	PD-M	–	DC-M DQ-M PD-M	–	–
Data-Out Received	–	–	–	–	–	–	–	DC-O DQ-O	–	–	–	–	–	–	–	–	DC-M DQ-M PD-M	DC-O DQ-O PD-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

PR = Power on protocol
NL = Nexus loss protocol
DM = Device management protocol
AC = All command protocols
ND = Non-data command protocol
PI = PIO data-in and PIO data out command protocols
DC = PIO data-in, PIO data-out, DMA data-in, & DMA data-out protocols
DQ = DMA queued command protocol
AP = All Packet command protocols
PN = Packet non-data command protocol
PD = Packet data command protocols

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 10)
(argument 1)	represents one or more arguments that are mandatory for the transport protocol service or data transfer protocol service (see table 11)
[argument 2]	represents one or more arguments that are optional for the transport protocol service or data transfer protocol service (see table 11)\
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 protocol

5.2.1.1 Power-on protocol overview

A transport-specific power-on event causes a device to perform its device-specific internal diagnostics routine, and, upon completion of that routine, return information to the host. Because power may not be applied to the host and the device at the same time, separate protocols are shown for the host and the device. However, an ATA domain does not come into existence until the host and at least one device are powered on and capable of communicating with each other through the service delivery subsystem (see 4.2.1).

5.2.1.2 Host power-on protocol description

Figure 11 shows the processing of the host power-on procedure call.

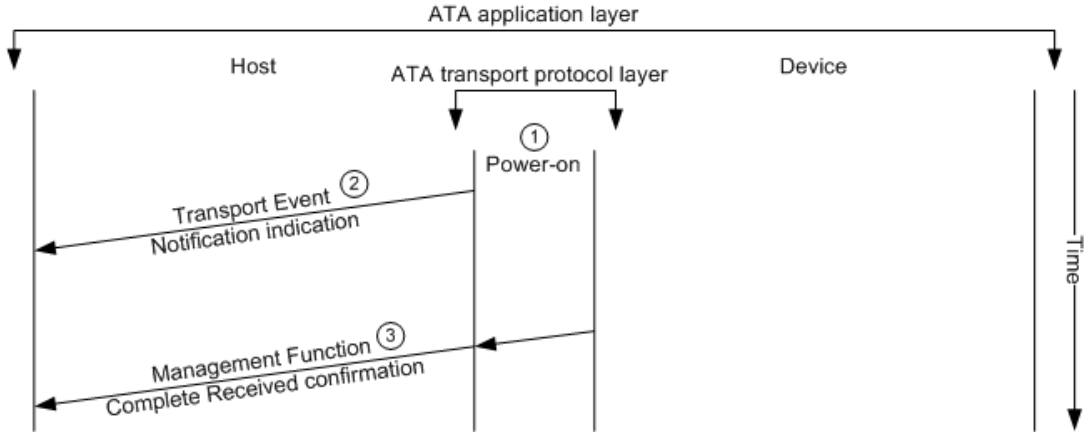

Figure 11 — Host Power-on procedure call

The following is the description of the sequence for the host power-on procedure call:

1) A transport-specific power-on reset event occurs. The power-on reset event detected by the host may be different from the power-on reset event detected by the device;
2) The host port sends a Transport Event Notification (Power-on/Hardware Reset) indication to the application client;
3) After the host port receives a response from a device port (see 5.2.1.3), the host port sends a Management Function Complete Received ([Device], Status, [Error]) confirmation to the application client. This confirmation may be stimulated by vendor-specific communication between the application client and the host port.

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 returns the information that indicates the current state of the features).

The content of the Device and Power-on/Hardware Reset arguments are as defined in 5.1.3.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

5.2.1.3 Device power-on protocol description

Figure 12 shows the processing of the device power-on procedure call.

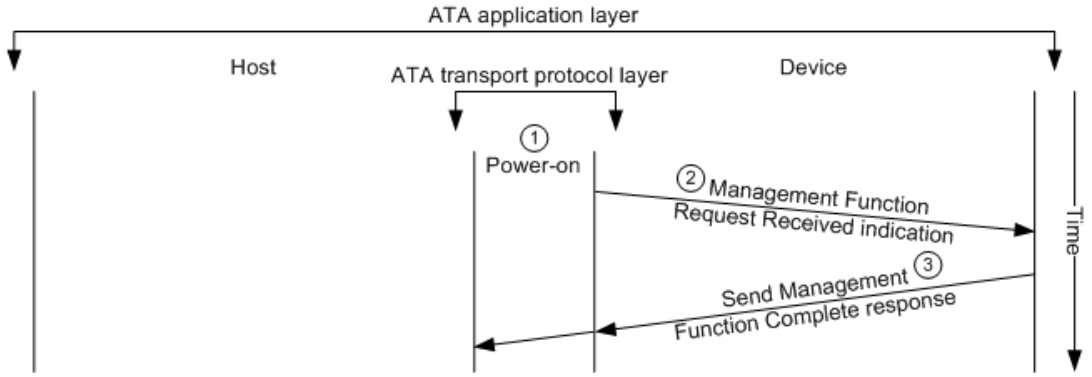

Figure 12 — Device power-on procedure call

The following is the description of the sequence for the device power-on procedure call:

1) A transport-specific power-on reset event occurs. The power-on reset event detected by the host may be different from the power-on reset event detected by the device;
2) The device port sends a Management Function Request Received (Power-on/Hardware Reset) indication to the device server, causing the device server to process a device-specific hardware reset and internal diagnostics routine and other transport-specific operations (e.g., determining the presence of device 0 (see ATA8-APT)); and
3) The device server sends a Send Management Function Complete (Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem.

The content of the Device and Power-on/Hardware Reset arguments are as defined in 5.1.3.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

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 power-on procedure call.

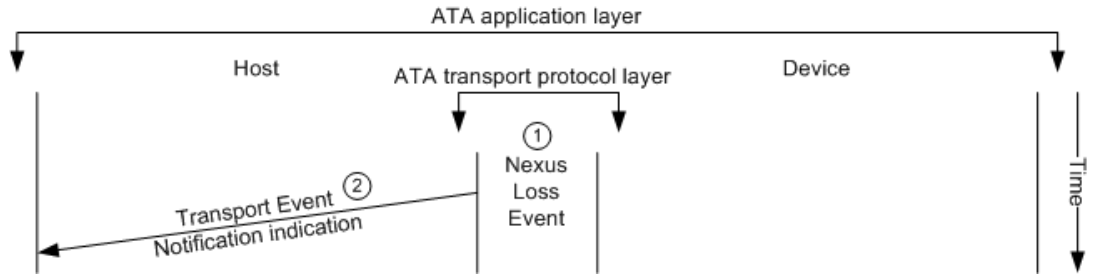

Figure 13 — Nexus loss procedure call

The following is the description of the sequence for the nexus loss procedure call:

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

The content of the Device and Nexus Loss arguments are as 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) Hard reset;
b) Software Reset; and
c) Setting the interrupt state.

One and only one device management function shall be requested by the host in each device management sequence.

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 text and numbers in figure 14 show conditional or optional behavior as described in this subclause.

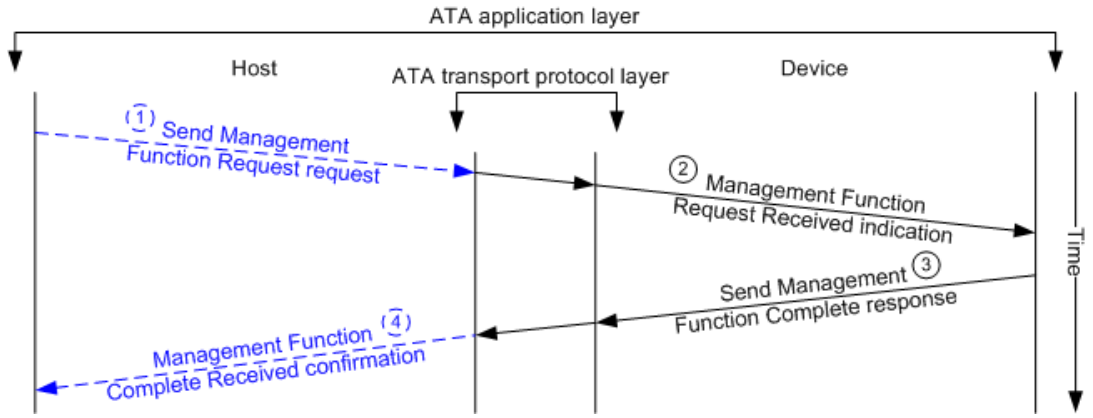

Figure 14 — Device management procedure call

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 ([Power-on/Hardware Reset], [Software Reset], [Enable Interrupts], [Disable Interrupts], [Device]) request to the host port causing the host port to transmit the request via the service delivery subsystem; or
   B) The host port generates a Send Management Function Request ([Power-on/Hardware Reset], [Software Reset], [Enable Interrupts], [Disable Interrupts]) request and transmits the request via the service delivery subsystem;

   One and only one of the arguments (i.e., Power-on/Hardware Reset, Software Reset, Enable Interrupts, or Disable Interrupts) is included in a Send Management Function Request.
2) The device port sends a Management Function Request Received ([Power-on/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 Power-on/ Hardware Reset, then the device server processes its device-specific hardware reset and internal diagnostics routine and other transport-specific operations;
   B) If the argument included in the Management Function Request Received indication is Software Reset, then the device server processes its device-specific software reset and internal diagnostics routine and other transport-specific operations;
   C) If the argument included in the Management Function Request Received indication is Enabled 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) The device server sends a Send Management Function Complete (Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem; and
4) If the device management function was initiated by the application client, then 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 sequence if either the Power-on/Hard 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 returns the information that indicates the current state of the features).

The content of the Device, Power-on/Hardware Reset, Software Reset, Enable Interrupts, and Disable Interrupts arguments are as defined in 5.1.3.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

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 transport 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 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.

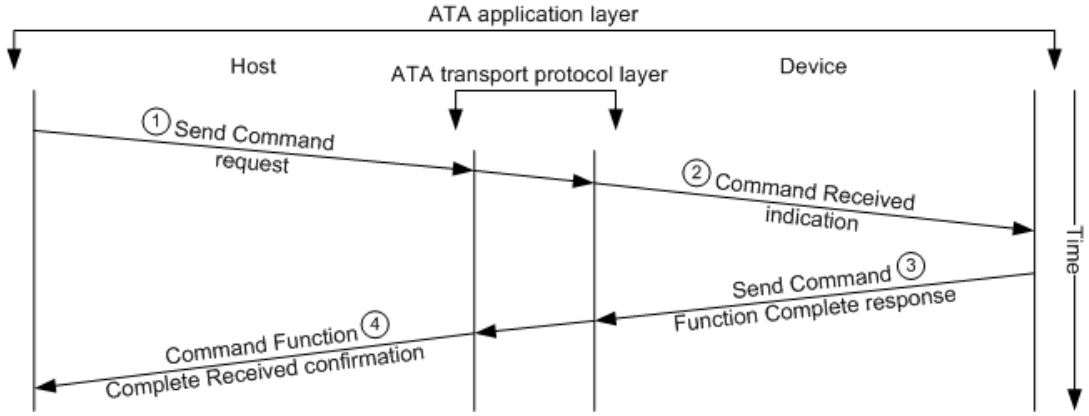

Figure 15 — Non-data command procedure call

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 transmit the request via the service delivery subsystem;
2) The device port receives the Send Command request and sends a Command Received (Command, [LBA], [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 ([LBA], Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem; and
4) The host port sends a Command Function Complete Received ([Device], [LBA], Status, [Error]) confirmation to the application client.

The content of the Command, Device, LBA, Count, and Features arguments are as defined in 5.1.3.

If the LBA, Count, and/or Features arguments contain command-specific information, then that information is defined in the ATA8-ACS standard.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

Receipt of an EXECUTE DEVICE DIAGNOTICS command or a DEVICE RESET command may cause a device server to perform one or more device-specific actions (e.g., perform a device-specific internal diagnostics routine) and/or one or more transport-specific actions before sending a Send Command Function Complete response to the device port. See the ATA8-ACS, ATA8-APT, and ATA8-AST standards for more details about the actions taken by the device upon receipt of these commands.

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 includes the transfer of one or more DRQ data blocks (see 3.1.12) from the device to the host. The PIO data-in command protocol is differentiated from the DMA data-in command protocol by returning status before transferring each DRQ data block, as opposed to transferring status after all data for the command is transferred. 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 text and numbers in figure 16 show conditional or optional behavior as described in this subclause.

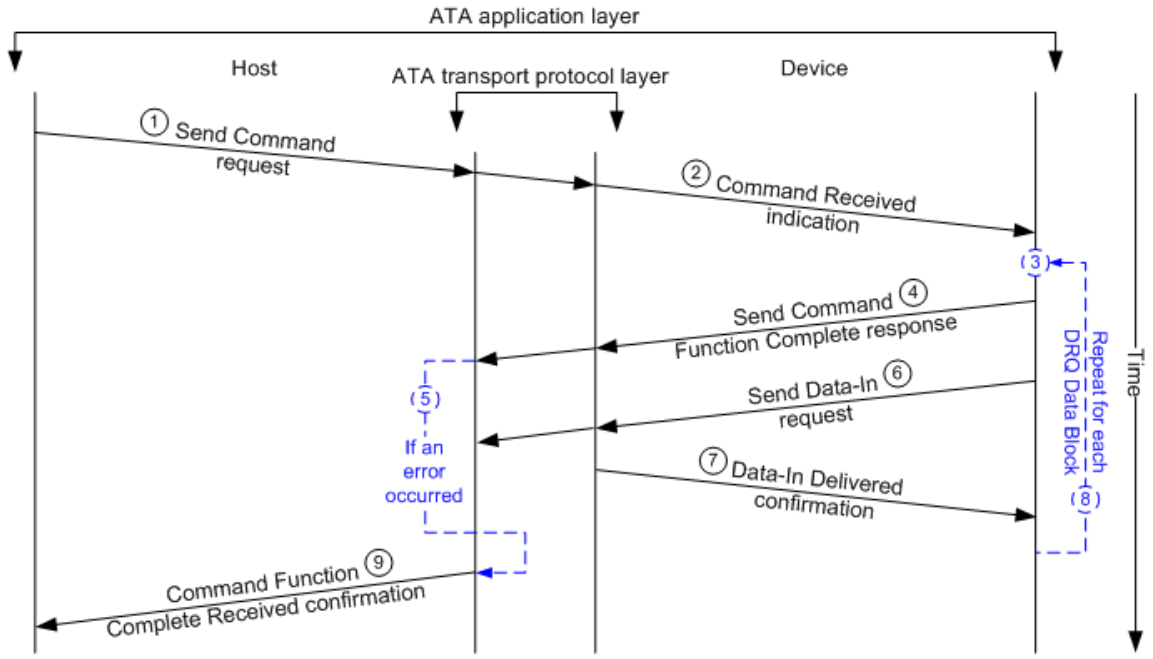

Figure 16 — PIO data-in command procedure call

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 transmit the request via the service delivery subsystem;
2) The device port receives the request and 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 there is no error in the Command Received indication, then the device server fetches the data for the command;
4) The device server sends a Send Command Function Complete ([LBA], Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem;

5) If the Error argument was present in the Send Command Function Complete response, then the device server may stop transmitting data and go to step 9;
6) 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 available the data via the service delivery subsystem. If this is not the first DRQ data block for the command, then Host Buffer contains the value from the previous Host Buffer plus the number of bytes transferred 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;
7) The device port transmits data to the host and sends a Data-In Delivered ([LBA], Status, [Error]) confirmation to the device server indicating that data was delivered to the host or that an error occurred while attempting to deliver the data;
8) If additional data transfers are required for the command, then go to step 3; and
9) When all of the data has been transferred for the command or an error occurred during the operation, 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 status has been sent to the host but before completion of the data transfer for the command.

The content of the Command, Device, LBA, Count, Features, DRQ Data Block, Host Buffer, and Device Buffer arguments are as defined in 5.1.3.

If the LBA, Count, and/or Features arguments contain command-specific information, then that information is defined in the ATA8-ACS standard.

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).

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

5.3.4 PIO data-out command protocol

5.3.4.1 PIO data-out command protocol overview

Processing a PIO data-out command includes the transfer of one or more DRQ data blocks from the host to the device. The PIO data-out command protocol is differentiated from other data-out command protocols by returning status after transferring each DRQ data block, as opposed to transferring status after all data for the command is transferred. 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-in command procedure call. The blue dashed lines and adjacent text and numbers in figure 17 show conditional or optional behavior as described in his subclause.

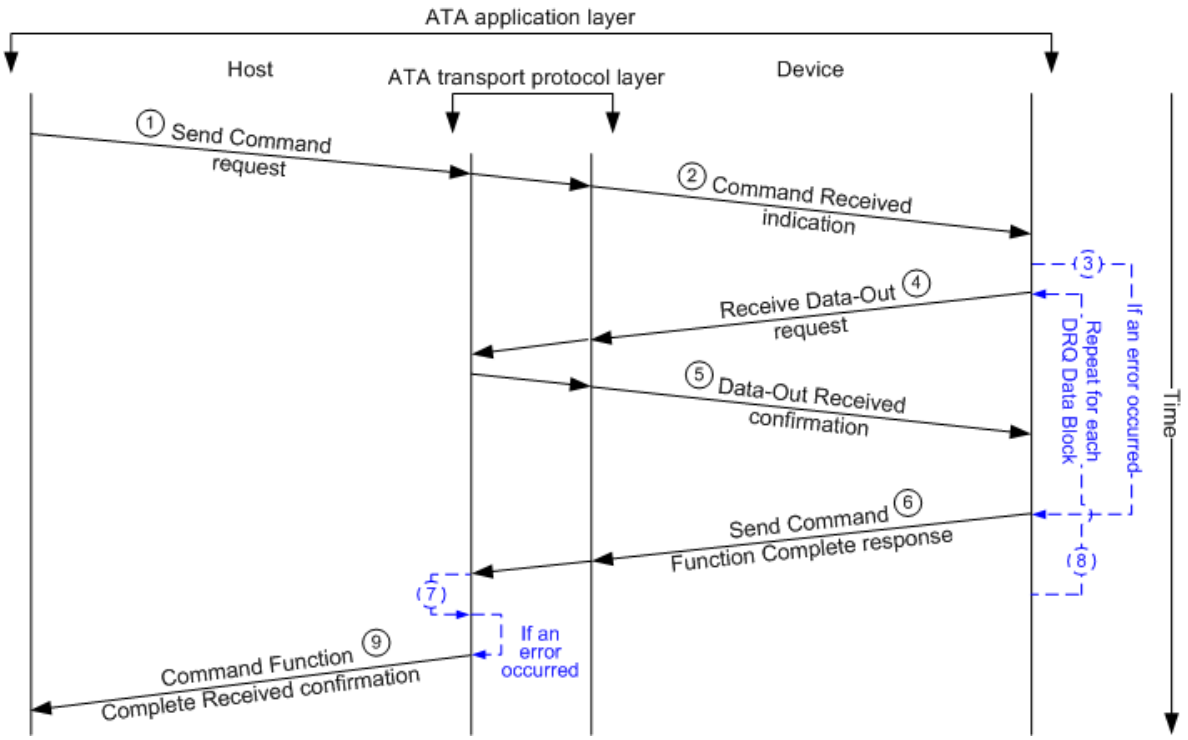

Figure 17 — PIO data-out command procedure call

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 transmit the request via the service delivery subsystem;
2) The device port receives the request and 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 there is an error in the Command Received indication, 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 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) The device port receives data from the host and sends a Data-Out Received ([LBA], Status, [Error]) confirmation to the device server indicating that data was received from the host or that an error occurred while attempting to receive the data;
6) The device server sends a Send Command Function Complete ([LBA], Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem;
7) If the Error argument was present in the Send Command Function Complete response, then the host port may stop sending data and go to step 9;
8) If additional data transfers are required for the command, then go to step 4; and
9) The host port sends a Command Function Complete Received ([Device], [LBA], Status, [Error]) confirmation to the application client.

The content of the Command, Device, LBA, Count, Features, DRQ Data Block, Host Buffer, and Device Buffer arguments are as defined in 5.1.3.

If the LBA, Count, and/or Features arguments contain command-specific information, then that information is defined in the ATA8-ACS standard.

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).

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

5.3.5 DMA data-in command protocol

5.3.5.1 DMA data-in command protocol overview

Processing of a DMA data-in command includes the transfer of data blocks from the device to the host. The DMA data-in command protocol is differentiated from the PIO data-in command protocol by returning status after all data is transferred or when an error occurs, as opposed to transferring status before transferring each DRQ data block. See the ATA8-ACS standard for description of the DMA data-in commands.

5.3.5.2 DMA data-in command protocol description

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

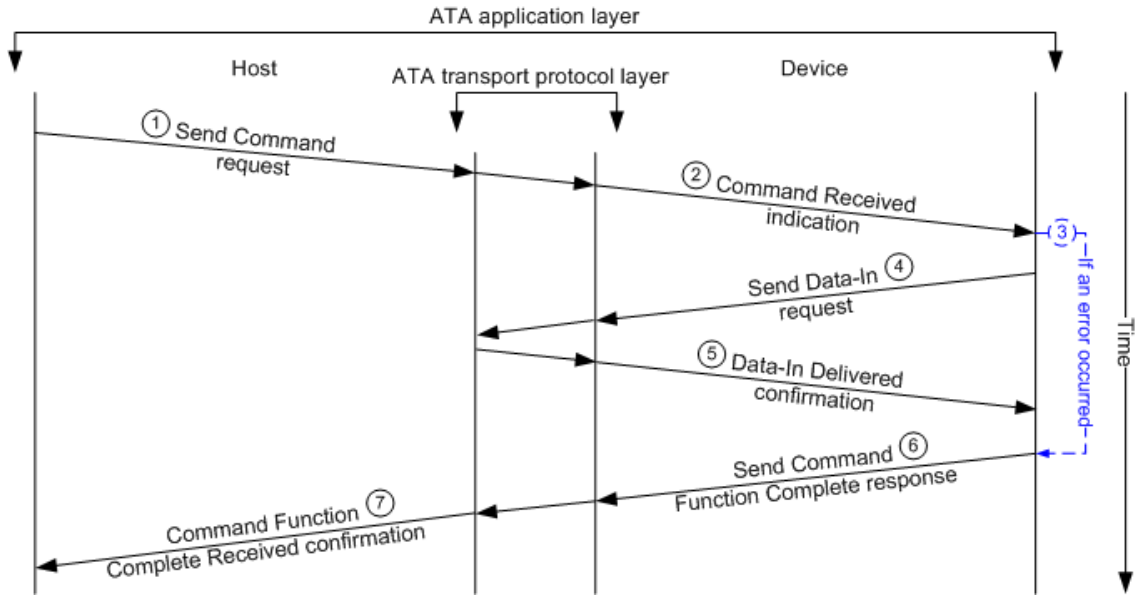

Figure 18 — DMA data-in command procedure call

The following is the description of the sequence of the DMA 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 transmit the request via the service delivery subsystem;
2) The device port receives the request and 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 there is an error in the Command Received indication, then go to step 6.
4) The device server fetches the data for the command and sends a Send Data-In (Count, [Host Buffer], Device Buffer) request to the device port causing the device port to make available the data via the service delivery subsystem;

5) The device port transmits data to the host and sends a Data-In Delivered ([LBA], Status, [Error]) confirmation to the device server indicating that data was delivered to the host or that an error occurred while attempting to deliver the data;
6) The device server sends a Send Command Function Complete ([LBA], Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem; and
7) The host port sends a Command Function Complete Received ([Device], [LBA], Status, [Error]) confirmation to the application client.

The content of the Command, Device, LBA, Count, Features, DRQ Data Block, Host Buffer, and Device Buffer arguments are as defined in 5.1.3.

If the Features argument contains command-specific information (e.g., for the READ STREAM DMA EXT command), then that information is defined in the ATA8-ACS standard.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

5.3.6 DMA data-out command protocol

5.3.6.1 DMA data-out command protocol overview

Processing of a DMA data-out command includes the transfer of data blocks from the host to the device. The DMA data-out command protocol is differentiated from the PIO data-out command protocol by returning status after all data is transferred or when an error occurs, as opposed to transferring status after transferring each DRQ data block. See the ATA8-ACS standard for description of the DMA data-out commands.

5.3.6.2 DMA data-out command protocol description

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

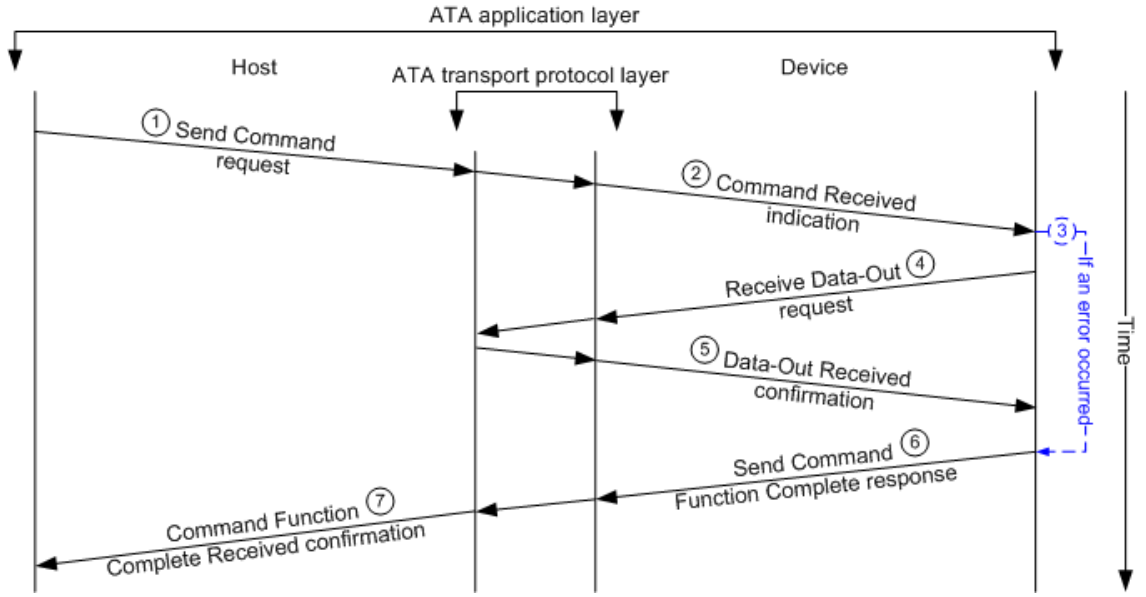

Figure 19 — DMA data-out command procedure call

The following is the description of the sequence of the DMA 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 transmit the request via the service delivery subsystem;

2) The device port receives the request and 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 there is an error in the Command Received indication, then go to step 6;
4) 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) The device port receives data from the host and sends a Data-Out Received ([LBA], Status, [Error]) confirmation to the device server indicating that data was received from the host or that an error occurred while attempting to receive the data;
6) The device server sends a Send Command Function Complete ([LBA], Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem; and
7) The host port sends a Command Function Complete Received ([Device], [LBA], Status, [Error]) confirmation to the application client.

The content of the Command, Device, LBA, Count, Features, DRQ Data Block, Host Buffer, and Device Buffer arguments are as defined in 5.1.3.

If the Features argument contains command-specific information (e.g., for the WRITE STREAM DMA EXT command), then that information is defined in the ATA8-ACS standard.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

5.3.7 DMA queued command protocol

5.3.7.1 DMA queued command protocol overview

Processing of a DMA queued command includes the transfer of data blocks from the device to the host or from the host to the device. The DMA queued command protocol is differentiated from the PIO data-in command protocol by returning status after all data is transferred or when an error occurs, as opposed to transferring status before transferring each DRQ data block.

The DMA queued command protocol is differentiated from the DMA data-in command protocol and the DMA data-out command protocol by the ability for the device, after receiving one DMA queued command, to signal the host that the device is not ready to transfer data for the command but is able to receive additional DMA queued commands. See the ATA8-ACS standard for description of the of DMA queued commands.

5.3.7.2 DMA queued command protocol description

Figure 20 shows the processing of the DMA queued command procedure call. The blue dashed lines and adjacent text and numbers in figure 20 show conditional or optional behavior as described in this subclause.

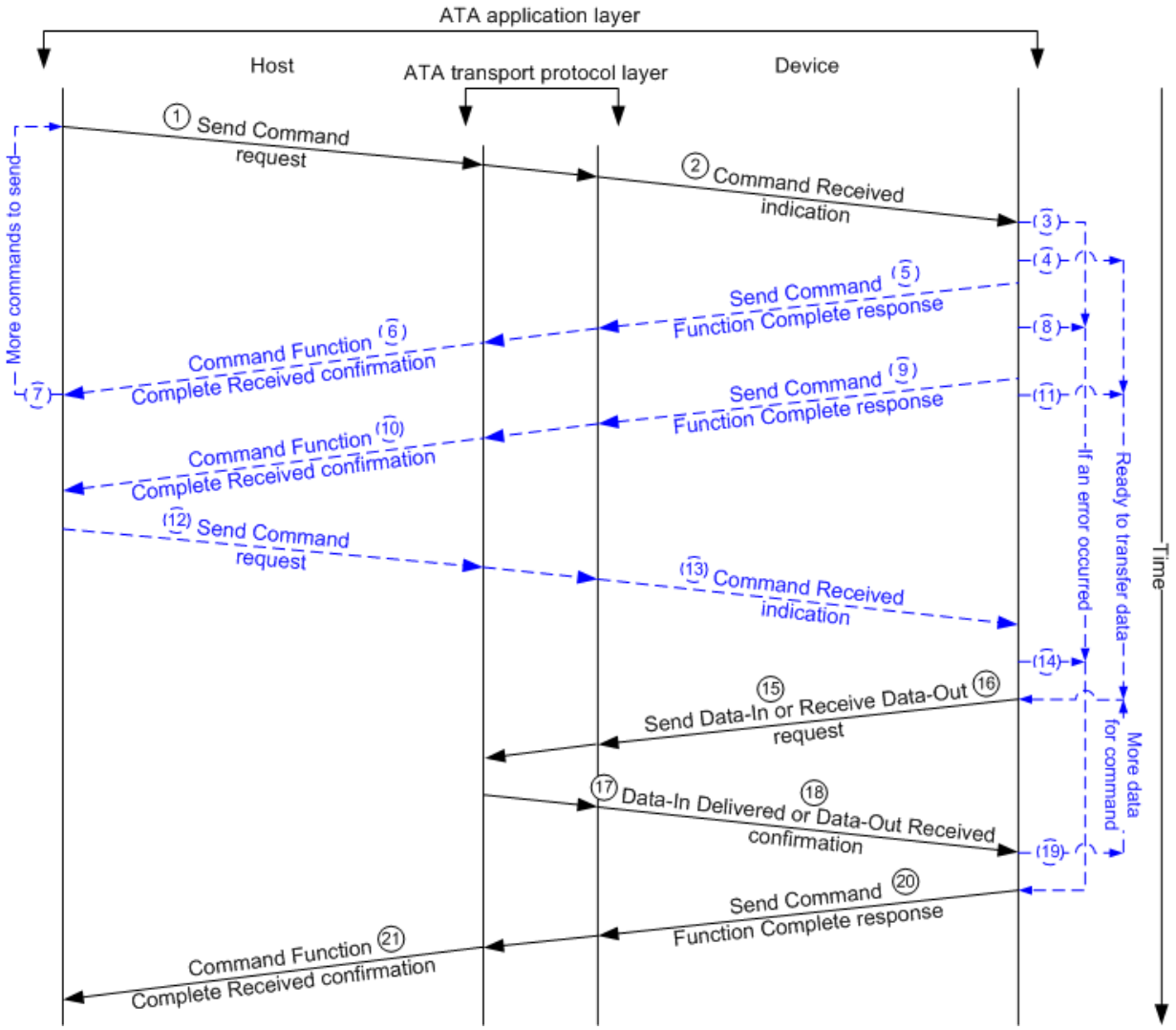

Figure 20 — DMA queued command procedure call

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 transmit the request via the service delivery subsystem;
2) The device port receives the request and 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 there is an error in the Command Received indication, 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 (Status) response to the device port causing the device port to make available the response 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) 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 (Status) response to the device port because the device server was not ready to transfer data for a queued command, and an error occurs for a queued command for which data has not been transferred, then go to step 20;
9) If the device server sent a Send Command Function Complete (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 (Tag, Status) response to the device port causing the device port to make available the response via the service delivery subsystem;
10) If the host port receives a Function Complete (Tag, Status) response via the service delivery subsystem, then the host port sends a Command Function Complete Received ([Device], Tag, Status) confirmation to the application client;
11) If the transport protocol does not require a command to initiate data transfer, then go to step 15;
12) If the transport protocol requires a command to initiate data transfer, 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 is ready to transmit or receive data, then the application client sends a Send Command (Command, [Device]) request to the host port causing the host port to transmit the request via the service delivery subsystem (the Command argument in this step contains the code for the SERVICE command (see ATA8-ACS));
13) The device port receives the request and sends a Command Received (Command) indication to the device server causing the device server to process the command;
14) If there is an error in the Command Received indication, then go to step 20;
15) If the data to be transferred is for a data-in command, then the device server sends a Send Data-In (Count, [Host Buffer], Device Buffer) request to the device port causing the device port to make available the data via the service delivery subsystem;
16) If the data to be transferred is for a data-out command, 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;
17) If the data to be transferred is for a data-in command, then the device port transmits data to the host and sends a Data-In Delivered ([LBA], Status, [Error]) confirmation to the device server indicating that data was delivered to the host or that an error occurred while attempting to deliver the data
18) If the data to be transferred is for a data-out command, then the device port receives data from the host and sends a Data-Out Received ([LBA], Tag, Status, [Error]) confirmation to the device server indicating that data was received from the host or that an error occurred while attempting to receive the data;
19) If additional data transfers are required for the command, then go to step 15;
20) The device server sends a Send Command Function Complete ([LBA], Tag, Status, [Error]) response to the device port causing the device port to make available the request via the service delivery subsystem; and
21) The host port sends a Command Function Complete Received ([Device], [LBA], Tag, Status, [Error]) confirmation to the application client.

The content of the Command, Device, LBA, Count, Features, Host Buffer, and Device Buffer arguments are as defined in 5.1.3.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

5.3.8 Packet command protocols

5.3.8.1 Packet command protocols overview

The PACKET command 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). The procedure call to be processed is determined by the content of the command packet:

a) If the command packet contains a non-data command, then the Packet non-data command procedure call is processed (see 5.3.8.2);
b) If the command packet contains a PIO data-in command, then the Packet PIO data-in command procedure call is processed (see 5.3.8.3);
c) If the command packet contains a PIO data-out command, then the Packet PIO data-out command procedure call is processed (see 5.3.8.4); or
d) If the command packet contains a DMA command, then the Packet DMA command procedure call is processed (see 5.3.8.5).

5.3.8.2 Packet non-data command protocol

5.3.8.2.1 Packet non-data command protocol overview

Processing of a Packet non-data command involves no data transfer.

5.3.8.2.2 Packet non-data command protocol description

Figure 21 shows the processing of the Packet non-data command procedure call.

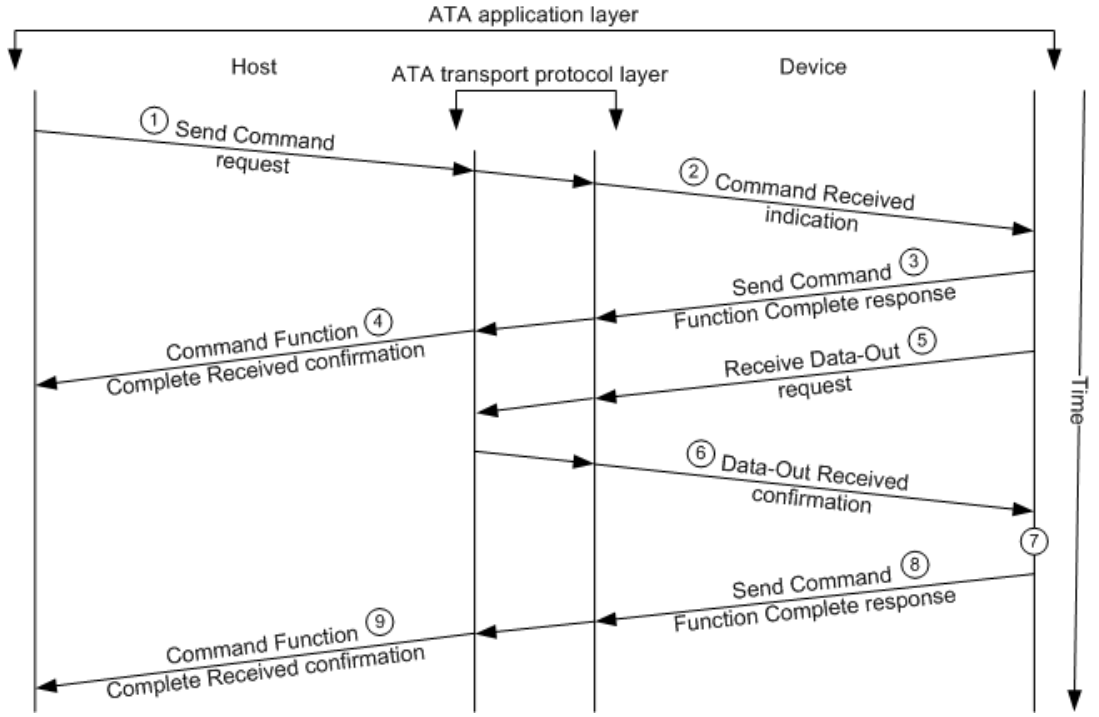

Figure 21 — Packet non-data command procedure call

The following is the description of the sequence of the Packet non-data command procedure call:

1) The application client sends a Send Command (Command, [Device], Count, [Features]) request to the host port causing the host port to transmit the request via the service delivery subsystem (the Command argument in this step contains the code for the PACKET command (see ATA8-ACS));
2) The device port receives the request and 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, Input/Output, Command/Data, Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem (in this step, the Input/Output argument contains zero, and the Command/Data argument contains one (see ATA8-ACS));

4) The host port sends a Command Function Complete Received ([Device], Count, Status, [Error]) confirmation to the application client;
5) If there was no error in the Command Received indication, then 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;
6) The device port receives data from the host and sends a Data-Out Received (Status, [Error]) confirmation to the device server indicating that data was received from the host or that an error occurred while attempting to receive the data;
7) The device server parses the command received during the data transfer and processes the command;
8) 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 available the response via the service delivery subsystem (in this step the Input/Output argument contains one and the Command/Data argument contains one); and
9) The host port sends a Command Function Complete Received ([Device], Status, [Error]) confirmation to the application client.

The content of the Command, Device, LBA, Count, and Features arguments are as defined in 5.1.3.

Command-specific information contained in the Features argument is defined in the ATA8-ACS standard.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

5.3.8.3 Packet PIO data-in command protocol

5.3.8.3.1 Packet PIO data-in command protocol overview

Processing of a Packet PIO data-in command includes the transfer of one or more DRQ data blocks (see 3.1.12) from the device to the host. The Packet PIO data-in command protocol is differentiated from the Packet DMA command protocol by returning status before transferring each DRQ data block, as opposed to transferring status after all data for the command is transferred.

5.3.8.3.2 Packet PIO data-in command description

Figure 22 shows the processing of the Packet PIO data-in command procedure call. The blue dashed lines and adjacent text and numbers in figure 22 show conditional or optional behavior as described in this subclause.

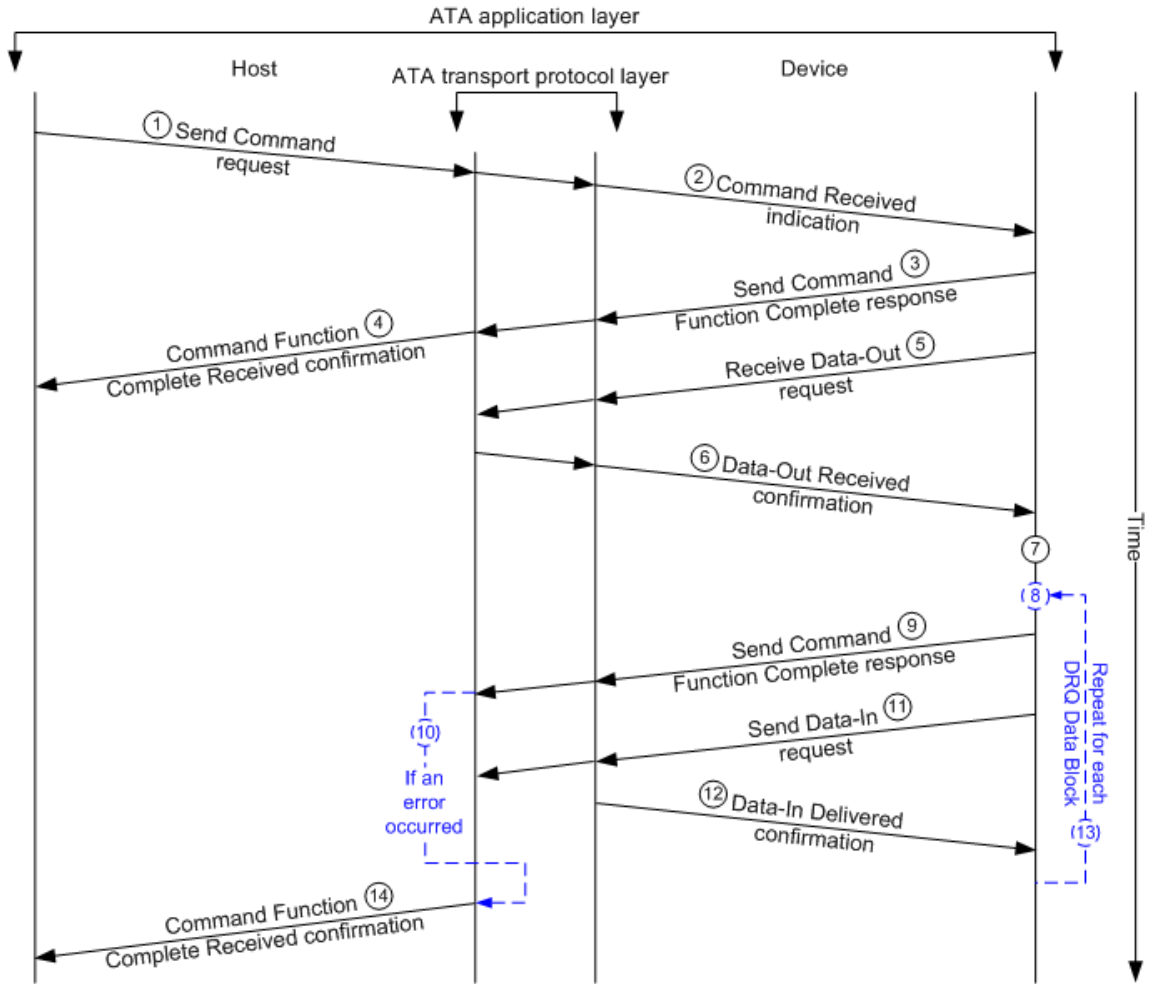

Figure 22 — Packet PIO data-in command procedure call

The following is the description of the sequence of the Packet data-in command procedure call:

1) The application client sends a Send Command (Command, [Device], Count, [Features]) request to the host port causing the host port to transmit the request via the service delivery subsystem (the Command argument in this step contains the code for the PACKET command (see ATA8-ACS));
2) The device port receives the request and 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, Input/Output, Command/Data, Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem (in this step, the Input/Output argument contains zero, and the Command/Data argument contains one (see ATA8-ACS);
4) The host port sends a Command Function Complete Received ([Device], Count, Status, [Error]) confirmation to the application client;
5) If there was no error in the Command Received indication, then 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;
6) The device port receives data from the host and sends a Data-Out Received (Status, [Error]) confirmation to the device server indicating that data was received from the host or that an error occurred while attempting to receive the data;

7) The device server parses the command received during the data transfer;
8) If there is no error in the parsed command, then the device server fetches the data for the command;
9) The device server sends a Send Command Function Complete (Count, Input/Output, Command/Data, Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem (in this step the Input/Output argument contains one and the Command/Data argument contains zero);
10) If the Error argument was present in the Send Command Function Complete response, then the device server may stop transmitting data and go to step 14;
11) 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 available the data via the service delivery subsystem;
12) The device port transmits data to the host and sends a Data-In Delivered (Status, [Error]) confirmation to the device server indicating that data was delivered to the host or that an error occurred while attempting to deliver the data;
13) If additional data transfers are required for the command, then go to step 7; and
14) When all of the data has been transferred for the command or an error occurred during the operation, the host port sends a Command Function Complete Received ([Device], Status, [Error]) confirmation to the application client.

The content of the Command, Device, LBA, Count, Features, DRQ Data Block, Host Buffer, and Device Buffer arguments are as defined in 5.1.3.

Command-specific information contained in the Features argument is defined in the ATA8-ACS standard.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

5.3.8.4 Packet PIO data-out command protocol

5.3.8.4.1 Packet PIO data out-command protocol overview

Processing a Packet PIO data-out command includes the transfer of one or more DRQ data blocks from the host to the device. The Packet PIO data-out command protocol is differentiated from the Packet DMA command protocol by returning status after transferring each DRQ data block, as opposed to transferring status after all data for the command is transferred.

5.3.8.4.2 Packet PIO data-out command protocol description

Figure 23 shows the processing of the Packet PIO data-out command procedure call. The blue dashed lines and adjacent text and numbers in figure 23 show conditional or optional behavior as described in this subclause.

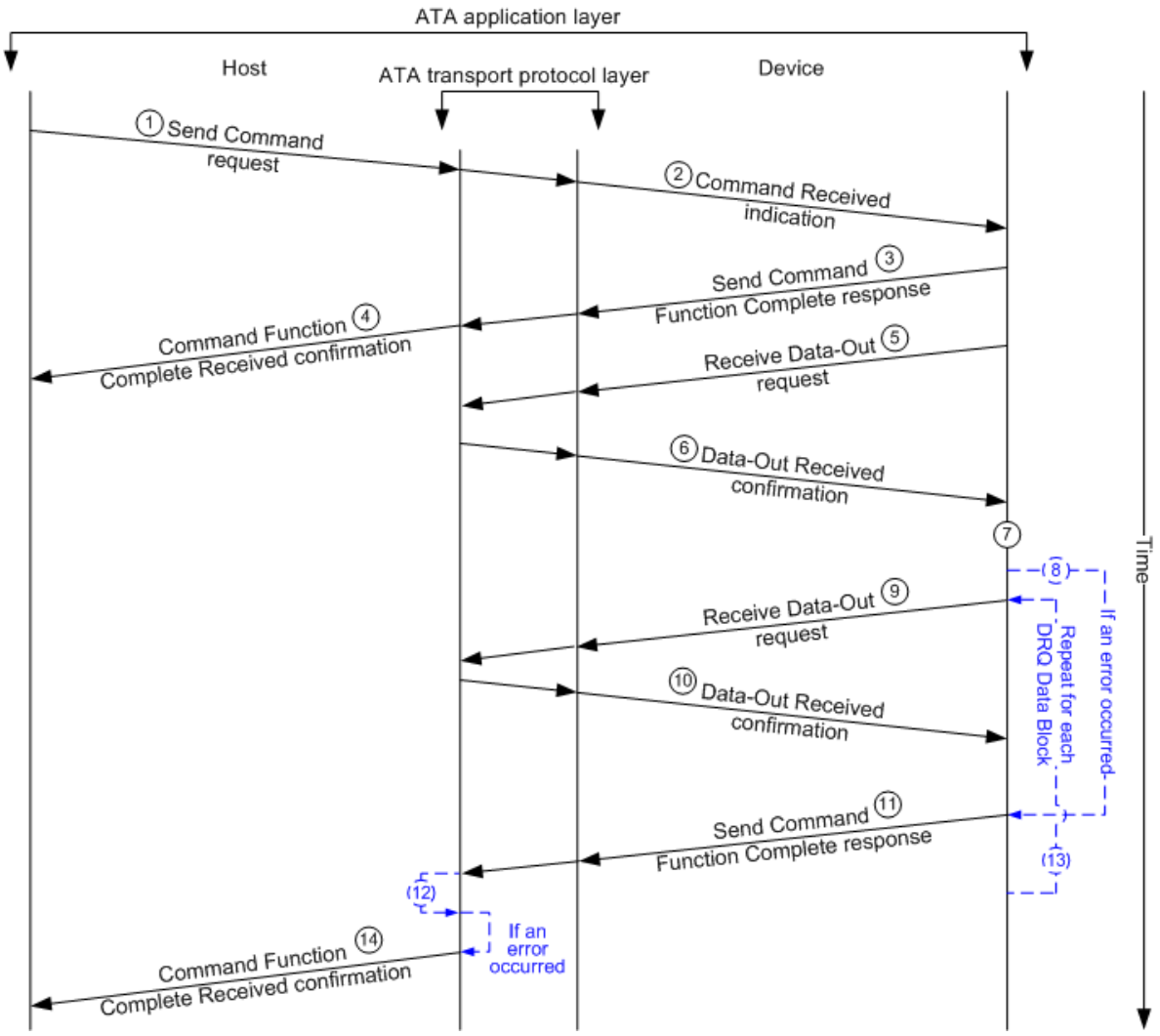

Figure 23 — Packet PIO data-out command procedure call

The following is the description of the sequence of the Packet data-in command procedure call:

1) The application client sends a Send Command (Command, [Device], Count, [Features]) request to the host port causing the host port to transmit the request via the service delivery subsystem (the Command argument in this step contains the code for the PACKET command (see ATA8-ACS));
2) The device port receives the request and 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, Input/Output, Command/Data, Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem (in this step, the Input/Output argument contains zero, and the Command/Data argument contains one (see ATA8-ACS);
4) The host port sends a Command Function Complete Received ([Device], Count, Status, [Error]) confirmation to the application client;

5) If there was no error in the Command Received indication, then 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;
6) The device port receives data from the host and sends a Data-Out Received (Status, [Error]) confirmation to the device server indicating that data was received from the host or that an error occurred while attempting to receive the data;
7) The device server parses the command received during the data transfer;
8) If there is an error in the parsed command, then go to step 11;
9) 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;
10) The device port receives data from the host and sends a Data-Out Received (Status, [Error]) confirmation to the device server indicating that data was received from the host or that an error occurred while attempting to receive the data;
11) The device server sends a Send Command Function Complete (Count, Input/Output, Command/Data, Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem (in this step the Input/Output argument contains one and the Command/Data argument contains zero);
12) If the Error argument was present in the Send Command Function Complete response, then the host port may stop sending data and go to step 14;
13) If additional data transfers are required for the command, then go to step 9; and
14) The host port sends a Command Function Complete Received ([Device], Status, [Error]) confirmation to the application client.

The content of the Command, Device, LBA, Count, Features, DRQ Data Block, Host Buffer, and Device Buffer arguments are as defined in 5.1.3.

If the LBA, Count, and/or Features arguments contain command-specific information, then that information is defined in the ATA8-ACS standard.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.

5.3.8.5 Packet DMA command protocol

5.3.8.5.1 Packet DMA command protocol overview

Processing of a Packet DMA command includes the transfer of data blocks from the device to the host. The Packet DMA command protocol is differentiated from the Packet PIO data-in command protocol and the Packet PIO data-out command protocol by returning status after all data is transferred or when an error occurs, as opposed to transferring status before or after transferring each DRQ data block.

5.3.8.5.2 Packet DMA command protocol description

Figure 24 shows the processing of the Packet DMA command procedure call.

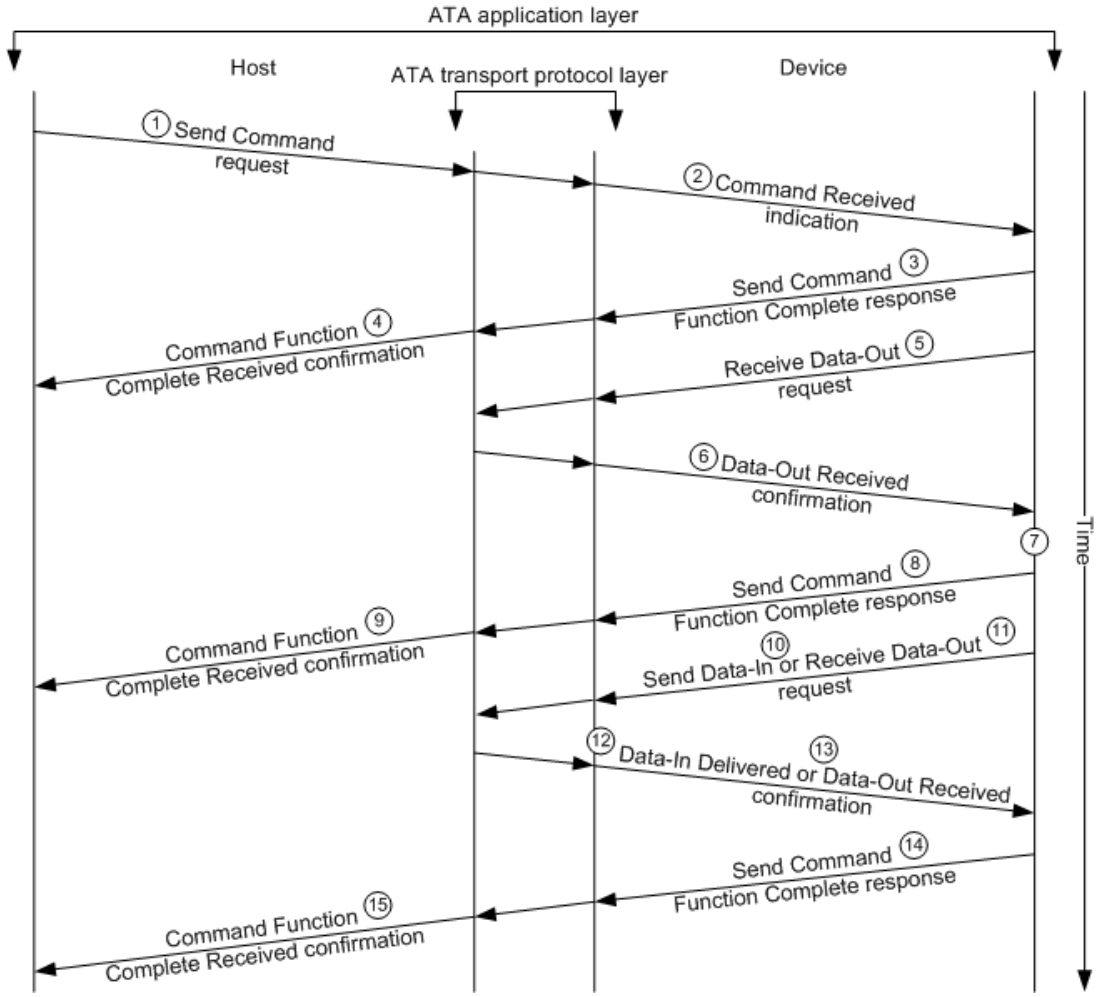

Figure 24 — Packet DMA command procedure call

The following is the description of the sequence of the Packet DMA command procedure call:

1) The application client sends a Send Command (Command, [Device], Count, [Features]) request to the host port causing the host port to transmit the request via the service delivery subsystem (the Command argument in this step contains the code for the PACKET command (see ATA8-ACS));
2) The device port receives the request and 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, Input/Output, Command/Data, Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem (in this step, the Input/Output argument contains zero, and the Command/Data argument contains one (see ATA8-ACS);
4) The host port sends a Command Function Complete Received ([Device], Count, Status, [Error]) confirmation to the application client;
5) If there was no error in the Command Received indication, then 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;
6) The device port receives data from the host and sends a Data-Out Received (Status, [Error]) confirmation to the device server indicating that data was received from the host or that an error occurred while attempting to receive the data;

7) The device server parses the command received during the data transfer;
8) The device server sends a Send Command Function Complete (Count, Input/Output, Command/Data, Status, [Error]) response to the device port causing the device port to make available the response via the service delivery subsystem (in this step:
    a) If the command is a data-in command, then the Input/Output argument contains one and the Command/Data argument contains zero; or
    b) If the command is a data-out command, then the Input/Output argument contains one and the Command/Data argument contains one; (see ATA8-ACS));
9) The host port sends a Command Function Complete Received ([Device], Status, [Error]) confirmation to the application client;
10) If there is data to be transferred for a data-in command, then the device server sends a Send Data-In (Count, [Host Buffer], Device Buffer) request to the device port causing the device port to make available the data via the service delivery subsystem;
11) If there is data to be transferred for a data-out command, 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;
12) If there is data to be transferred is for a data-in command, then the device port transmits data to the host and sends a Data-In Delivered (Status, [Error]) confirmation to the device server indicating that data was delivered to the host or that an error occurred while attempting to deliver the data;
13) If there is data to be transferred for a data-out command, then the device port receives data from the host and sends a Data-Out Received (Status, [Error]) confirmation to the device server indicating that data was received from the host or that an error occurred while attempting to receive the data;
14) 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 available the request via the service delivery subsystem (in this step, the Input/Output argument contains one and the Command/Data argument contains one); and
15) The host port sends a Command Function Complete Received ([Device], Status, [Error]) confirmation to the application client.

The content of the Command, Device, LBA, Count, Features, Host Buffer, and Device Buffer arguments are as defined in 5.1.3.

Command-specific information contained in the Features argument is defined in the ATA8-ACS standard.

The content of the Status and Error arguments are as defined in the ATA8-ACS standard.