[MS-AXDS]:
Auxiliary Display Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
09/25/2009	0.1	Major	First Release.
11/06/2009	1.0	Major	Updated and revised the technical content.
12/18/2009	1.0.1	Editorial	Revised and edited the technical content.
01/29/2010	1.0.2	Editorial	Revised and edited the technical content.
03/12/2010	1.1	Minor	Updated the technical content.
04/23/2010	1.1.1	Editorial	Revised and edited the technical content.
06/04/2010	1.2	Minor	Updated the technical content.
07/16/2010	2.0	Major	Significantly changed the technical content.
08/27/2010	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	2.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

The Auxiliary Display Protocol (AXDS) is a command-based packet system that provides a standard method for transferring data between Auxiliary Display Protocol-compatible devices and PC hosts.

Note The terms "AXDS device driver" and "AXDS driver" in this document refer to the AXDS driver running on the PC host.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

big-endian
globally unique identifier (GUID)
little-endian
Unicode

The following terms are specific to this document:

application: See **gadget**.

application/endpoint pair: The **application** must pair with the **endpoint** on the device. The **endpoint** advertises itself as supporting a data format.

AXDS device: See **gadget**.

capability negotiation: A connection initiator can determine whether the acceptor supports these connection types by sending the first message for the connection and determining the acceptor's level of support from the response.

endpoint: A software component on the device that advertises being able to handle a data format. An **endpoint** defines a data format. It also defines a set of events that can be sent back to the **application**; for example, user-input actions. An **endpoint** identifies itself with a **globally unique identifier (GUID)**.

gadget: A mini-application or a piece of code running on the PC host that sends data to devices using AXDS. The **gadget** retrieves data from a data source such as an application or Web service, and sends this data to AXDS, which sends it to the appropriate devices.

notification: An alert to the device user; normally implemented with a pop-up window with an optional alert sound associated with it. A common example is a meeting reminder.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site,

http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[BMP] Murray, J.D., "Graphics File Formats FAQ (Part 3 of 4): Where to Get File Format Specifications", http://www.faqs.org/faqs/graphics/fileformats-faq/part3/index.html

[IANAPORT] Internet Assigned Numbers Authority, "Port Numbers", November 2006, http://www.iana.org/assignments/port-numbers

[ISO-639.2-CODES] International Organization for Standardization, "Codes for Representation of Names of Languages", November 2008, http://www.loc.gov/standards/iso639-2/php/code_list.php

[ISO-3166-MA] International Organization for Standards 3166 Maintenance agency, "ISO's focal point for country codes", http://www.iso.org/iso/country_codes.htm

Note There is a charge to download the specification.

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MSDN-CDT] Microsoft Corporation, "Common Data Types", http://msdn.microsoft.com/en-us/library/aa505945.aspx

[MSDN-CULTURE-INFO] Microsoft Corporation, "CultureInfo Class", http://msdn.microsoft.com/en-us/library/system.globalization.cultureinfo.aspx

[MSDN-Date-Format] Microsoft Corporation, "Day, Month, Year, and Era Format Pictures", http://msdn.microsoft.com/en-us/library/dd317787(VS.85).aspx

[MSDN-DEVCAP] Microsoft Corporation, "Device Capabilities", http://msdn.microsoft.com/en-us/library/ms744040(VS.85).aspx

[MSDN-Time-Format] Microsoft Corporation, "Hour, Minute, and Second Format Pictures", http://msdn.microsoft.com/en-us/library/dd318148(VS.85).aspx

[MSDN-WDKUM] Microsoft Corporation, "Windows Driver Kit: Windows SideShow, User Models", http://msdn.microsoft.com/en-us/library/ff548117.aspx

[UNIV-DRV] Microsoft Corporation, "Using the Universal Driver for Windows SideShow", December 2008, http://www.microsoft.com/whdc/device/sideshow/Univ-Drv_Sideshow.mspx

1.3 Overview

The AXDS host standardizes on how AXDS-enabled devices connect to computers that are running Windows. An AXDS driver can send commands to an AXDS-compatible device in the form of a packet. Either the host or the **AXDS device** can start communications. The AXDS device returns a packet that indicates the success or failure of the command. Additionally, the AXDS device can send event information to the AXDS host. The types of events include cache-content-missing, **application**, and user-change events. The AXDS driver acknowledges receipt of the event from the AXDS device by responding with an acknowledgment packet.

The AXDS host supports custom input/output control codes (IOCTLs).

Windows Vista® operating system, Windows Server® 2008 operating system, Windows® 7 operating system, and Windows Server® 2008 R2 operating system include an AXDS-compatible implementation of the host driver that connects to AXDS-compatible devices over multiple transport types.<1> [UNIV-DRV]

1.4 Relationship to Other Protocols

The AXDS protocol has no dependency on any specific protocol. It is designed to sit on top of other protocols in order to send and receive data. The underlying protocol needs to be reliable in the sense that packet order is maintained and dropped packets are detected.

1.5 Prerequisites/Preconditions

Requires a PC host running Windows Vista® operating system, Windows Server® 2008 operating system, Windows® 7 operating system, or Windows Server® 2008 R2 operating system with Windows SideShow installed. Also requires an AXDS-compatible device.

1.6 Applicability Statement

The use of the AXDS protocol is applicable to an environment where communication is required between an AXDS-enabled host PC and an AXDS-compatible device.

1.7 Versioning and Capability Negotiation

The AXDS protocol does not perform **capability negotiation**. However the AXDS protocol does pass the versioning information implicitly through the Sync command.

1.8 Vendor-Extensible Fields

The AXDS protocol provides an extensibility mechanism that enables device manufacturers to send custom, device-specific commands to a device. See section 2.2.3.2, SendPassThrough.

1.9 Standards Assignments

The standards assignments are listed in the following table.

Parameter	Value	Reference
MS-SideShow	5360/TCP 5360/UDP	See [IANAPORT].
MS-S-SideShow	5361/TCP 5361/UDP	See [IANAPORT].

2 Messages

2.1 Transport

The AXDS protocol has no dependency on any specific protocol. It is designed to sit on top of other protocols in order to send and receive data. The underlying protocol needs to be reliable in the sense that packet order is maintained and dropped packets are detected.

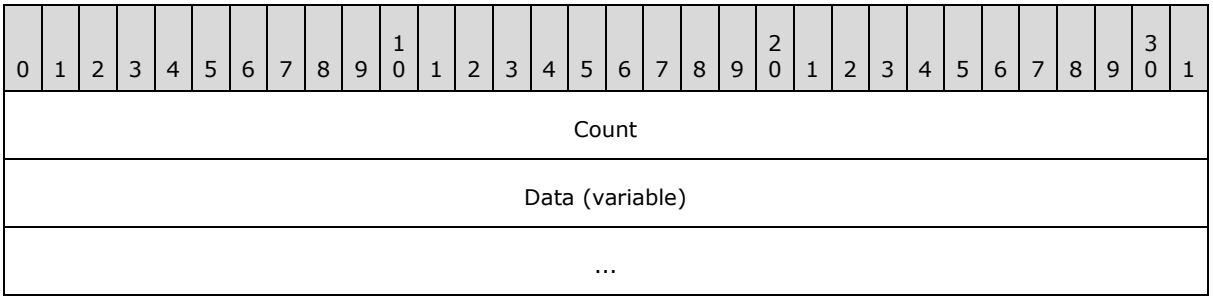
2.2 Message Syntax

2.2.1 Common Data Types

For additional information about the data types used in this document, see [MSDN-CDT].

2.2.1.1 Counted String

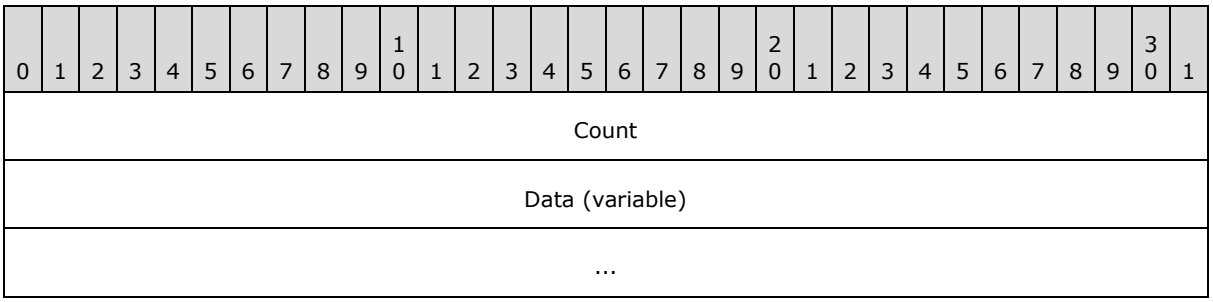
String arguments within packets are composed of a 4-byte value that indicates the length (in characters) of the string, followed by the string data. String data consists of 16-bit **Unicode** characters stored in **big-endian** byte order (bytes are numbered from left to right). Strings MUST NOT be null terminated.



Count (4 bytes): A DWORD that contains the number of elements in the Data field. If the count is zero, this indicates an empty string.

Data (variable): An array of 16-bit Unicode characters.

2.2.1.2 Counted Byte Array

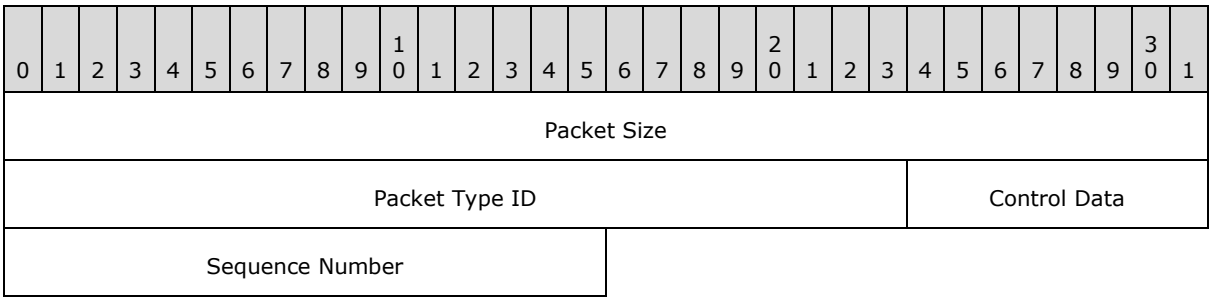


Count (4 bytes): A DWORD that contains the number of elements in the Data field. If the count is zero, this indicates an array with zero elements.

Data (variable): An array of bytes.

2.2.2 Auxiliary Display Protocol Packet Header

All command and data packets have a common header that contains the packet type, the sequence number, and the size of the packet. The following structure defines the common packet header.



Packet Size (4 bytes): A DWORD that denotes the size of the packet, in bytes, including the header and the payload.

Packet Type ID (3 bytes): An unsigned value that specifies the packet type. See section 2.2.2.2.

Control Data (1 byte): A BYTE where bit 24 must be set if the packet is a response packet to a command. Bit 25 must be set if the response is a NAK or cleared if the response is an ACK. Bits 26 through 31 are reserved for an error code, to be included only if bit 25 is set. See section 2.2.2.1.

Sequence Number (2 bytes): An unsigned SHORT integer that is incremented after each command packet. The first command packet will have a sequence number of zero. Each response packet will have the same sequence number. The same AXDS header definitions work with event packets. The sequence number is just another mechanism to match the command packet to the response packets. It is not to ensure the order of the command sequence.

2.2.2.1 Control Data

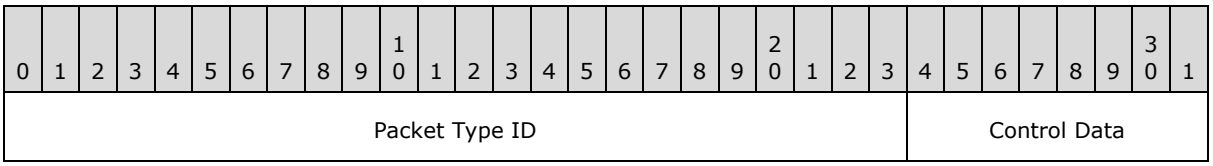
The packet type is a 4-byte value where the high byte is reserved for Control Data (**little-endian**). Bit 24 must be set if the packet is a response packet to a command. Bit 25 must be set if the response is a NAK or cleared if the response is an ACK. Bits 26 through 31 are reserved for an error code, to be included only if bit 25 is set.

ACK

If a received command is properly formed and has valid data, then the receiver MUST reply with an ACK. This is done by clearing bit 25 of the packet-type part of the packet header. Some ACK response packets may also contain the data that the command requests. The descriptions for the command packets include the specifics about the data that is sent in response to a particular command.

NAK

If a received command is not properly formed, has an unknown packet-type ID, or has invalid data, then the receiver should reply with a NAK. This is done by setting bit 25 of the packet-type part of the packet header. In addition, bits 26 through 31 can contain an error code for the failure of the command that is being responded to.



8-bit Control Data

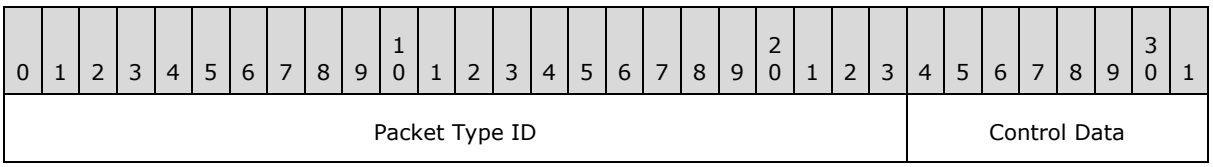
Bit 24: Response; set to 1 if the packet is a response to a command.

Bit 25: Response; set to 1 if the response is a NAK or set to 0 if the response is an ACK.

Bits 26-31: Reserved for any implementation-defined error code; to be included in transmission only if bit 25 is set.

2.2.2.2 Packet Type ID

The 24-bit Packet Type ID specifies the format of the data contained in the packet payload.



Bits 0-23: Denote a 24-bit Packet Type ID.

2.2.3 Command Packets

Command packets represent the operations that the AXDS platform can request the device to perform. All command and response data packets have a common header that contains the packet type, the packet number, and the size of the packet (section 2.2.2). The AXDS driver running on the host PC always initiates command packets. Command packets can be composed of several pieces of data that constitute arguments for the command. For every command, the device must return a response packet.

Note Whenever this document uses the terms "AXDS device driver" or "AXDS driver", it is referring to the AXDS driver running on the PC host.

The following sections specify command packet types.

2.2.3.1 Ping

This command provides a way for the host PC to test a connection with the AXDS device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000001. Both the devices and the device drivers can use a Ping command packet to test the connection. For example, see the Ping Event (section 2.2.4.1.1) for details about an AXDS device pinging the PC host.

2.2.3.1.1 Ping Command

The Auxiliary Display Protocol (AXDS) packet header (section 2.2.2) with Packet Type ID set to 0x000001.

2.2.3.1.2 Ping Response

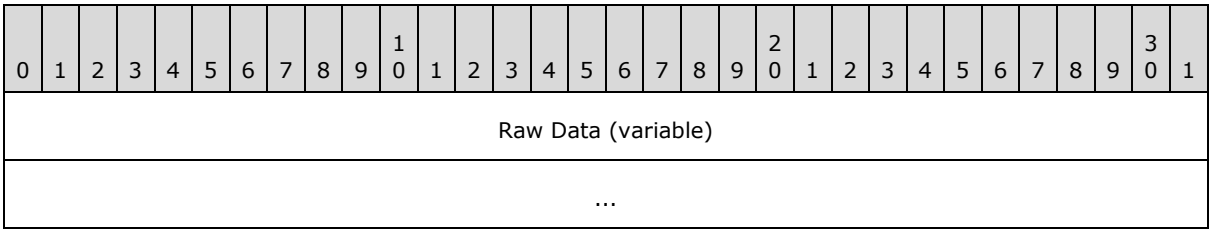
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.2 SendPassThrough

This command provides an extensibility mechanism that enables device manufacturers to define methods for sending custom, device-specific commands to a device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000002.

2.2.3.2.1 SendPassThrough Command

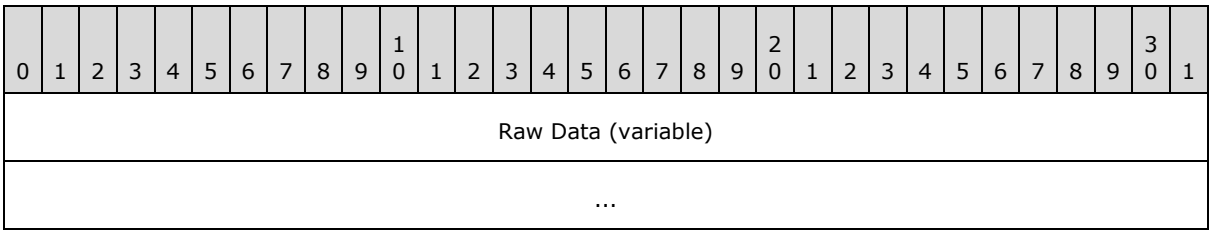
Following the AXDS Packet header, the command packet must contain the data in the following format, in the order shown below.



Raw Data (variable): A Counted Byte Array (section 2.2.1.2) that contains the data that can be sent across the wire.

2.2.3.2.2 SendPassThrough Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header followed by raw data, in bytes.



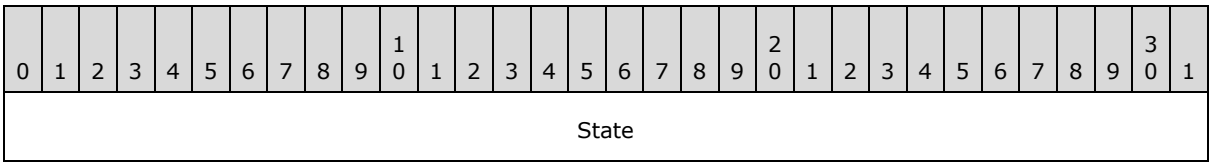
Raw Data (variable): A Counted Byte Array (section 2.2.1.2) that contains the data that can be sent across the wire.

2.2.3.3 Reset

This command restarts the device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000003.

2.2.3.3.1 Reset Command

Following the AXDS Packet header, the command packet must contain the data in the following format, in the order shown below.



State (4 bytes): A DWORD that indicates the desired state of the device once the command completes. The following values are possible.

Value	Meaning
0x00000000	Restart only.
0x00000001	Restart and wait in the boot-loader program.
Other	Manufacturer-defined values.

2.2.3.3.2 Reset Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.4 GetCurrentUser

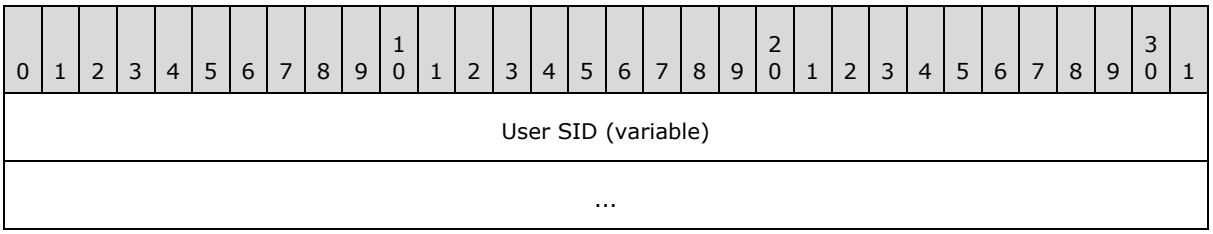
This command retrieves the security identifier (SID) in string format ([MS-DTYP] section 2.4.2) for the current user of the device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000101.

2.2.3.4.1 GetCurrentUser Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000101.

2.2.3.4.2 GetCurrentUser Response

Following the AXDS Packet header, the response packet must contain the data in the following format, in the order shown below.



User SID (variable): A Counted String (section 2.2.1.1) that contains the SID of the current user.

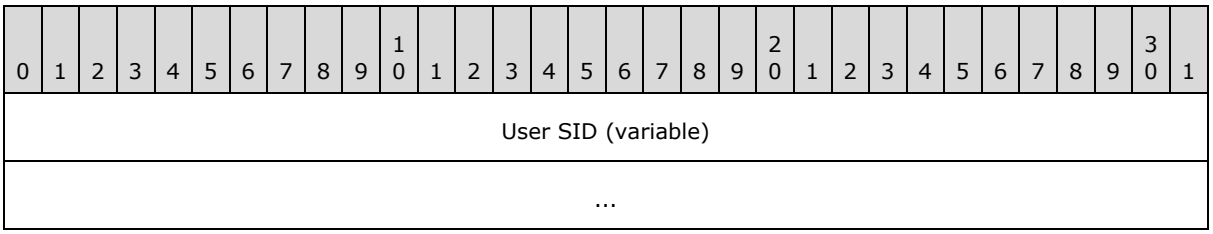
Note The SID for the current user in the response packet varies with the user model. For Assigned user model devices, a device should return the NULL SID (S-1-0-0) until it receives the current user's SID from a call to the SetCurrentUser command (section 2.2.3.5). If the device is using the Console user model, the driver should return the well-known interactive user's SID. See the following behavior note for more information about user models.<2>

2.2.3.5 SetCurrentUser

This command sets the current user of the device. The user must be identified by an SID. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000100.

2.2.3.5.1 SetCurrentUser Command

Following the AXDS Packet header, the command packet must contain the data in the following format, in the order shown below.



User SID (variable): A Counted String (section 2.2.1.1) that contains the SID of the current user.

2.2.3.5.2 SetCurrentUser Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

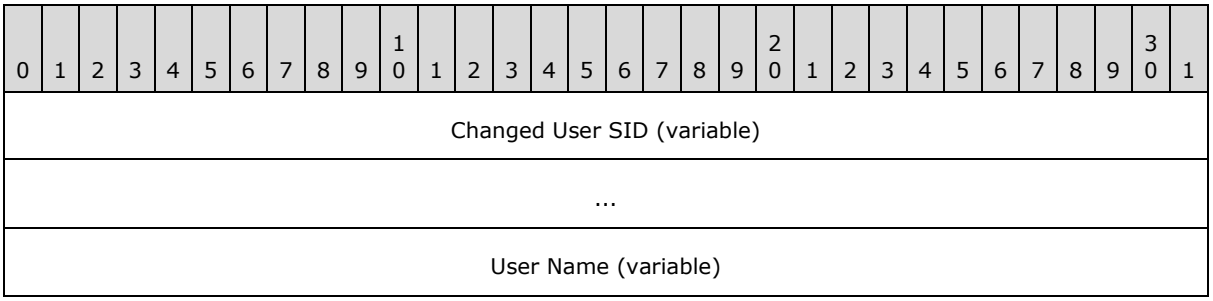
Note Send this command when a user becomes the current user of the device, which typically corresponds to a computer log-on. Send this command with a NULL SID when the current user is unknown, which typically corresponds to a computer log-off. See the following behavior note for more information about user models.<3>

2.2.3.6 SetUserState

This command notifies user-associated devices as to which users are available to be selected as the owner of the device. Users must be identified by SIDs. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000050. See the following behavior note for more information about user models.<4>

2.2.3.6.1 SetUserState Command

Following the AXDS Packet header, the command packet must contain the data in the following format, in the order shown below.



...
User State

Changed User SID (variable): A Counted String (section 2.2.1.1) that contains the SID of the user whose log-in state has been changed.

User Name (variable): A Counted String that contains the name of the user.

User State (4 bytes): A DWORD that indicates the availability of the specified user once the command completes. The following values are possible.

Value	Meaning
0x00000000	Available to be selected as owner of the device.
0x00000001	Unavailable to be selected as owner of the device.

2.2.3.6.2 SetUserState Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.7 GetDeviceFirmwareVersion

This command retrieves a string that identifies the version of the device firmware. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000102.

2.2.3.7.1 GetDeviceFirmwareVersion Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000102.

2.2.3.7.2 GetDeviceFirmwareVersion Response

Following the AXDS Packet header, the response packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Firmware Version (variable)																															
...																															

Firmware Version (variable): A Counted String (section 2.2.1.1) that contains the version of the firmware.

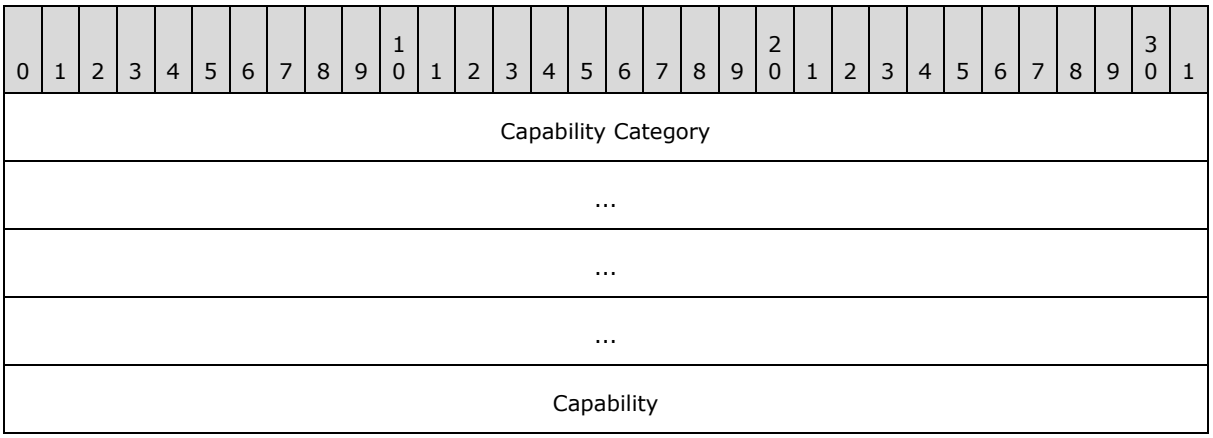
Note The implementer defines the format for the firmware version string.

2.2.3.8 GetCapabilities

This command retrieves a device capability value. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000103.

2.2.3.8.1 GetCapabilities Command

Following the AXDS Packet header, the command packet must contain the data in the following format, in the order shown below.

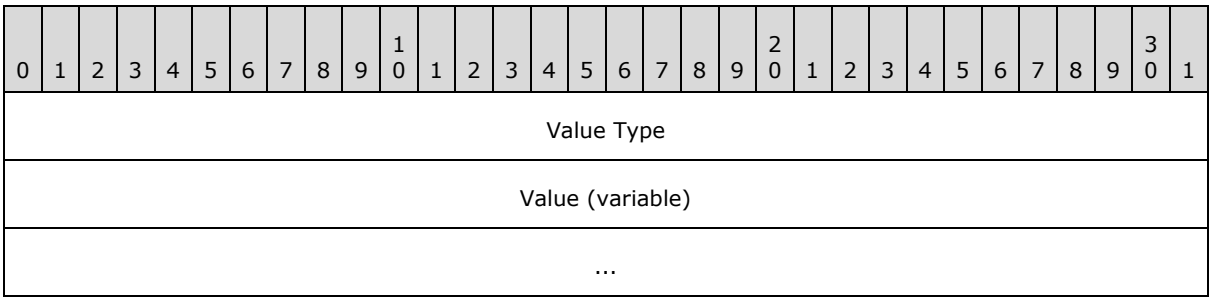


Capability Category (16 bytes): A **globally unique identifier (GUID)**, as defined in [MS-DTYP] section 2.3.2.2, that specifies the category of the requested capability.

Capability (4 bytes): A DWORD that specifies the requested capability.<5>

2.2.3.8.2 GetCapabilities Response

Following the AXDS Packet header, the response packet must contain the data in the following format, in the order shown below.



Value Type (4 bytes): A DWORD that indicates the type of the Value field. The supported **PROPVARIANT** types are VT_EMPTY, VT_I2, VT_I4, VT_R4, VT_R8, VT_DATE, VT_BOOL, VT_I1, VT_UI1, VT_UI2, VT_UI4, VT_I8, VT_UI8, VT_INT, VT_UINT, VT_LPWSTR, VT_CLSID, and VT_UI1 | VT_VECTOR.

Value (variable): The returned value of the requested capability.

Note The **PROPVARIANT**s must be encoded so that the first 4 bytes is the **VARTYPE** and the remaining bytes contain the data. For information about standard platform capabilities, see the following behavior note.<6> See section 10 for supported types.

2.2.3.9 GetApplicationOrder

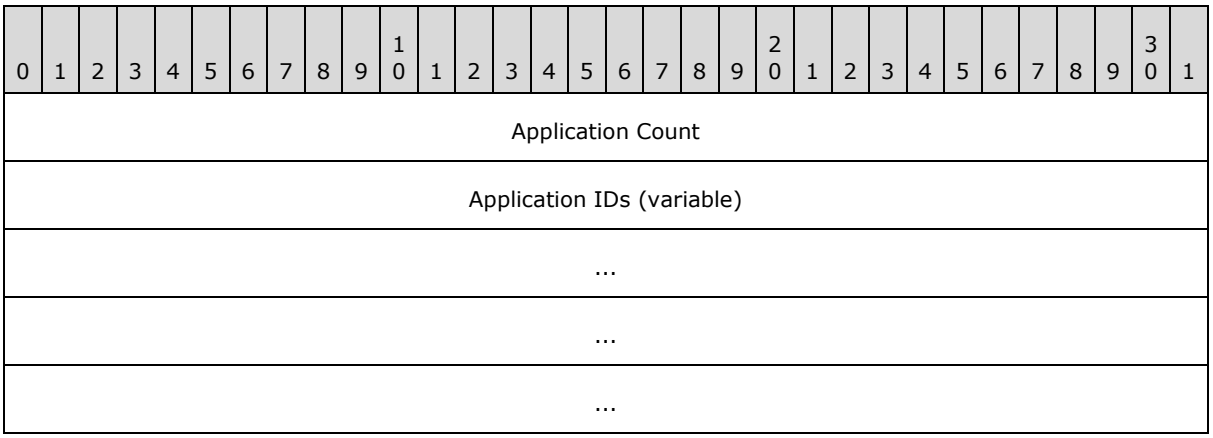
This command retrieves a display-ordered list of AXDS applications that have been added to the device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000104.

2.2.3.9.1 GetApplicationOrder Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000104.

2.2.3.9.2 GetApplicationOrder Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



Application Count (4 bytes): A DWORD that indicates the number of Application IDs returned.

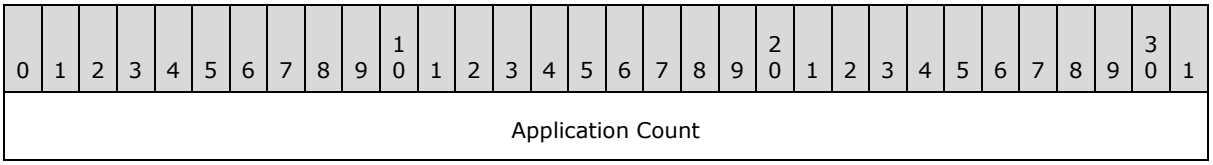
Application IDs (variable): An array of GUIDs that specifies the applications that are currently registered with the device.

2.2.3.10 SetApplicationOrder

This command specifies the display order for AXDS applications that are running on the device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000105.

2.2.3.10.1 SetApplicationOrder Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Application IDs (variable)
...
...
...

Application Count (4 bytes): A DWORD that indicates the number of Application IDs being sent to the device.

Application IDs (variable): An array of GUIDs that specifies the applications.

2.2.3.10.2 SetApplicationOrder Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.11 SetLanguage

This command specifies the current language and font size for a device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000106.

Note The GetCapabilities mechanism (section 2.2.3.8) is used to retrieve the language.

2.2.3.11.1 SetLanguage Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Language Info (variable)																															
...																															

Language Info (variable): A Counted String (section 2.2.1.1) that specifies the culture name (see [ISO-639.2-CODES] and [ISO-3166-MA]) along with the font size. Further information on the use of culture names can be found in [MSDN-CULTURE-INFO].

Note The device manufacturer must specify which languages and countries the device supports.

The language that this command specifies must be one of the set of languages that the GetCapabilities command (section 2.2.3.8.1) returns for the SIDESHOW_CAPABILITY_SUPPORTED_LANGUAGES capability. When the AXDS device first boots, the default language is set by the OEM (Device Manufacturer).

The format of the command string is

<culture_code>:n

where *n* is a 1-based index value for the set of supported font sizes.

Font sizes are indexed from smallest to largest. The font size index must be less than or equal to the number of fonts that the GetCapabilities command returns for the SIDESHOW_CAPABILITY_SUPPORTED_LANGUAGES capability.

2.2.3.11.2 SetLanguage Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.12 GetPreEnabledApplications

This command retrieves a list of applications, specified by the device, to be enabled by default for all users of the computer. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000107.

2.2.3.12.1 GetPreEnabledApplications Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000107.

2.2.3.12.2 GetPreEnabledApplications Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Count																															
Application and Endpoint Pair (variable)																															
...																															
...																															
...																															
...																															
...																															
...																															
...																															

Count (4 bytes): A DWORD that contains the total number of GUIDs. It does not contain the number of pairs.

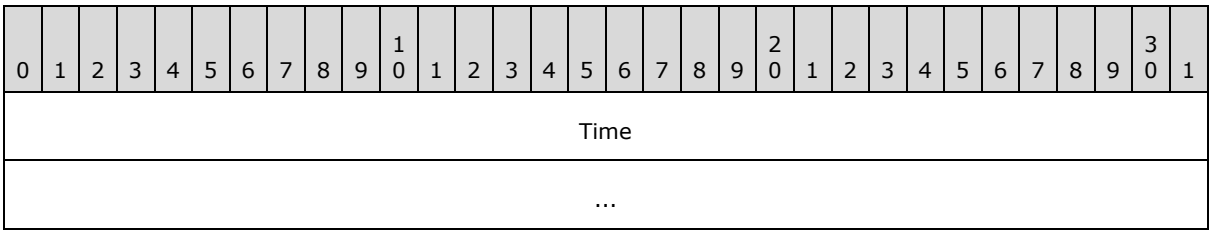
Application and Endpoint Pair (variable): An array of GUID pairs where the first GUID of each pair is an application ID and the second GUID of each pair is the **endpoint** type to which the application connects.

2.2.3.13 SetTime

This command specifies the current Coordinated Universal Time (UTC). The Packet Type ID field (section2.2.2.2) in the packet header is set to 0x000108.

2.2.3.13.1 SetTime Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Time (8 bytes): The current Coordinated Universal Time (UTC), which is formatted as a FILETIME structure ([MS-DTYP] section 2.3.1).

2.2.3.13.2 SetTime Response

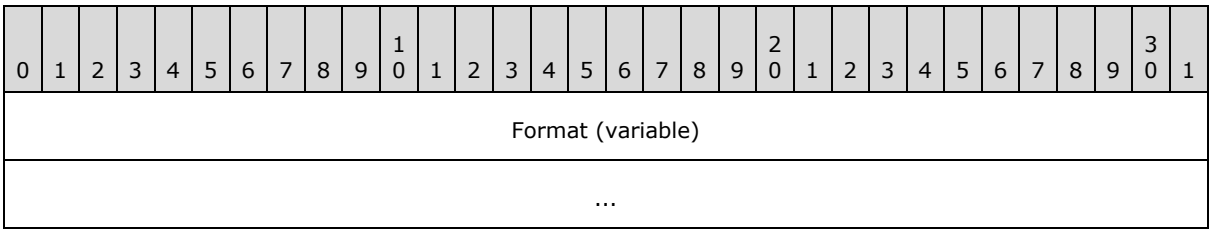
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.14 SetShortDateFormat

This command specifies the format for displaying a concise date. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000109.

2.2.3.14.1 SetShortDateFormat Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Format (variable): A Counted String (section 2.2.1.1) that specifies the pattern for displaying concise dates. For more information about date and time formats see section 8.

2.2.3.14.2 SetShortDateFormat Response

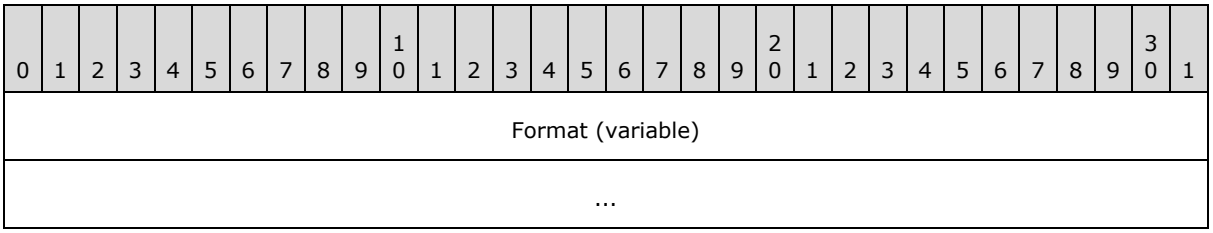
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.15 SetLongDateFormat

This command indicates the format for displaying a full date. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00010A.

2.2.3.15.1 SetLongDateFormat Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Format (variable): A Counted String (section 2.2.1.1) that specifies the pattern for displaying full dates. For more information about date and time formats see section 8.

2.2.3.15.2 SetLongDateFormat Response

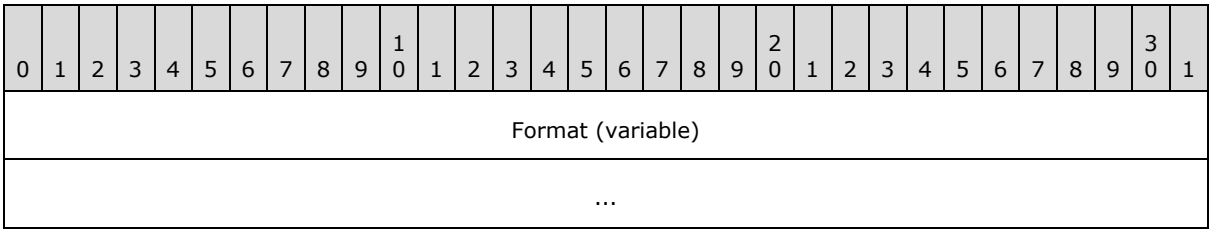
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.16 SetShortTimeFormat

This command specifies the format for displaying a concise time. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00010B.

2.2.3.16.1 SetShortTimeFormat Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Format (variable): A Counted String (section 2.2.1.1) that specifies the pattern for displaying concise times. For more information about date and time formats see section 8.

2.2.3.16.2 SetShortTimeFormat Response

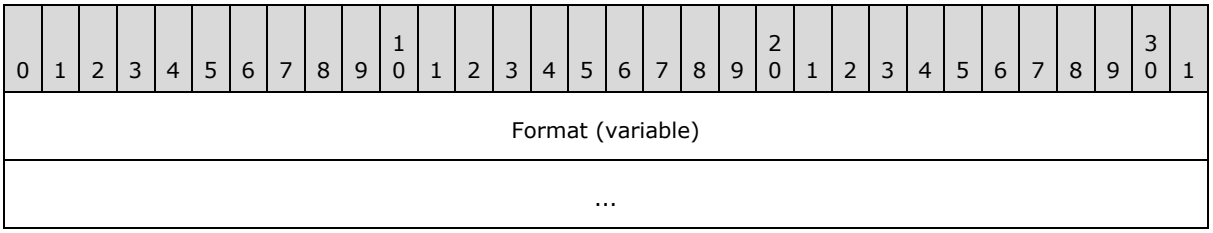
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.17 SetLongTimeFormat

This command indicates the format for displaying a full time. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00010C.

2.2.3.17.1 SetLongTimeFormat Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Format (variable): A Counted String (section 2.2.1.1) that specifies the pattern for displaying full times. For more information about date and time formats see section 8.

2.2.3.17.2 SetLongTimeFormat Response

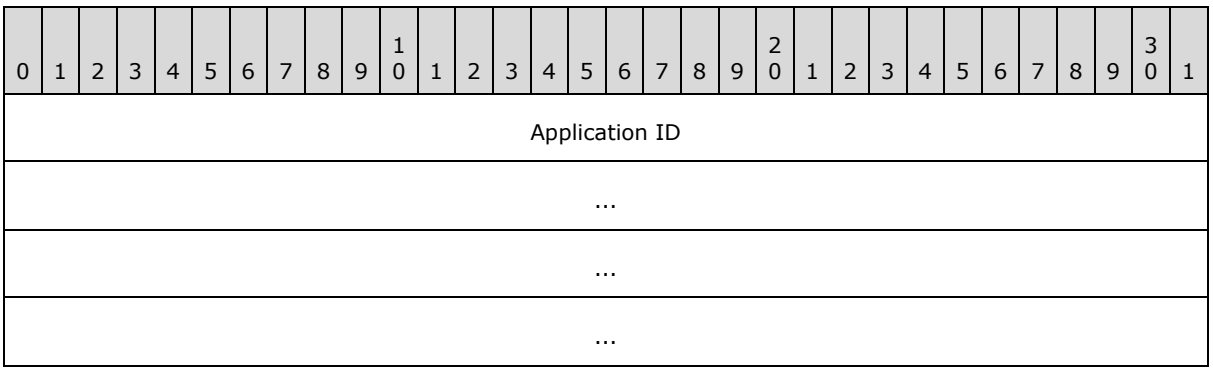
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.18 AddApplication

This command adds a new application to the AXDS device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00010D. This command enables the device to run an application--once the application is added to the device, the device can receive content.

2.2.3.18.1 AddApplication Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Endpoint ID
...
...
...
Application Name (variable)
...
Cache Policy
Online Only Setting
Large Icon (variable)
...
Medium Icon (variable)
...
Small Icon (variable)
...

Application ID (16 bytes): A GUID that identifies the application.

Endpoint ID (16 bytes): A GUID that identifies the endpoint to which the application connects. For a list of platform-defined Endpoint IDs, see the following behavior note.<7>

Application Name (variable): A Counted String (section 2.2.1.1) that specifies the display name.

Cache Policy (4 bytes): A DWORD value that determines how the cache behaves when it becomes full. Content may need to be removed from the cache if new content arrives for a **gadget** that is below its guaranteed limit and the cache is currently full.

Value	Description
0x00000000	Keep Newest (the default). Give priority to the newest content in the cache.
0x00000001	Keep Oldest. Give priority to the oldest content in the cache.
0x00000002	Keep Frequently Accessed. Give priority to the most frequently accessed content. Can be combined with Keep Oldest to give priority to the oldest item when multiple items have the same access frequency.

Value	Description
0x00000004	Keep Recently Accessed. Give priority to the content that has most recently been accessed by the user. Can be combined with Keep Oldest to give priority to the oldest item when multiple items have the same last access time.

Note The AXDS platform does not require implementers to use a cache or a specific cache algorithm; this information is included as a recommendation.

Online Only Setting (4 bytes): A Boolean value that specifies whether the application can be used only when the device is in communication with the PC host. The following table lists possible values for the Online Only Setting.

Value	Meaning
0x00000000	False
0x00000001	True (online only)

Large Icon (variable): A Counted Byte Array (section 2.2.1.2) containing a 48x48 pixel bitmap image [BMP] of the application icon.

Medium Icon (variable): A Counted Byte Array containing a 32x32 pixel bitmap image of the application icon.

Small Icon (variable): A Counted Byte Array containing a 16x16 pixel bitmap image of the application icon.

2.2.3.18.2 AddApplication Response

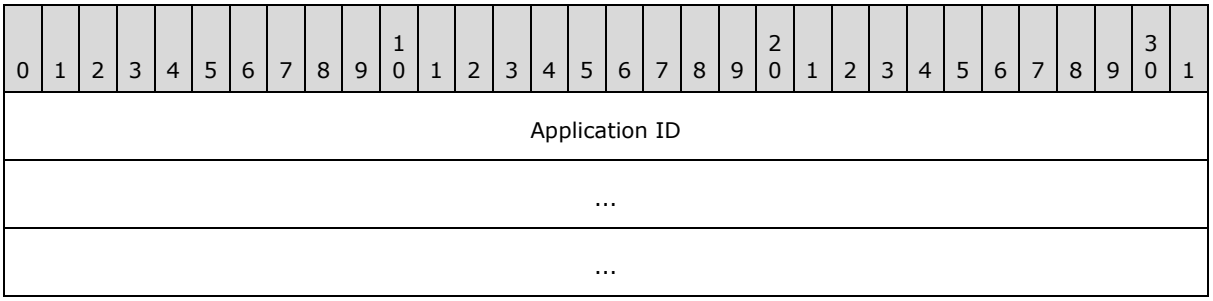
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.19 DeleteApplication

This command removes a previously added application from the device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00010E.

2.2.3.19.1 DeleteApplication Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



...

Application ID (16 bytes): A GUID that identifies the application.

2.2.3.19.2 DeleteApplication Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

Note This command will not remove preinstalled applications from the device.

2.2.3.20 DeleteAllApplications

This command removes all previously added applications from the device, the content for the applications, and their **notifications**. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00010F.

2.2.3.20.1 DeleteAllApplications Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x00010F.

2.2.3.20.2 DeleteAllApplications Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

Note This command must not remove preinstalled applications from the device.

2.2.3.21 AddNotification

This command adds a notification that is associated with the specified application. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000110.

2.2.3.21.1 AddNotification Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Application ID																															
...																															
...																															
...																															
Notification ID																															

Expiration Time
...
Notification Title (variable)
...
Notification Message (variable)
...
Notification Icon (variable)
...

Application ID (16 bytes): A GUID that identifies the application.

Notification ID (4 bytes): A DWORD that assigns an identifier to the notification.

Expiration Time (8 bytes): The time at which the notification expires. The value is formatted as a FILETIME structure ([MS-DTYP] section 2.3.1).

Notification Title (variable): A Counted String (section 2.2.1.1) that specifies the title of the notification.

Notification Message (variable): A Counted String that specifies the message text of the notification.

Notification Icon (variable): A Counted Byte Array (section 2.2.1.2) that contains a bitmap image [BMP] to be displayed with the notification.

2.2.3.21.2 AddNotification Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.22 DeleteNotification

This command removes a notification from the specified application. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000111.

2.2.3.22.1 DeleteNotification Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Application ID																															

...
...
...
Notification ID

Application ID (16 bytes): A GUID that identifies the application that created the notification.

Notification ID (4 bytes): A DWORD that identifies the notification to be deleted.

2.2.3.22.2 DeleteNotification Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.23 DeleteAllNotifications

This command removes all notifications from the specified application. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000112.

2.2.3.23.1 DeleteAllNotifications Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Application ID																															
...																															
...																															
...																															

Application ID (16 bytes): A GUID that identifies the application that created the notifications.

2.2.3.23.2 DeleteAllNotifications Response

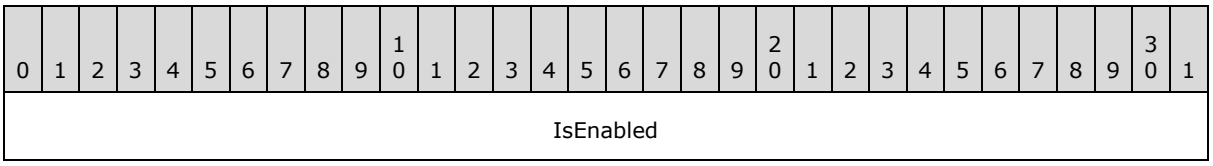
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.24 SetNotificationsEnabled

This command specifies whether notifications are enabled. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000113.

2.2.3.24.1 SetNotificationsEnabled Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



IsEnabled (4 bytes): A Boolean value that specifies the enabled state of the notifications. The following table lists possible values for IsEnabled.

Value	Meaning
0x00000000	False (notifications disabled)
0xFFFFFFFF	True (notifications enabled)

2.2.3.24.2 SetNotificationsEnabled Response

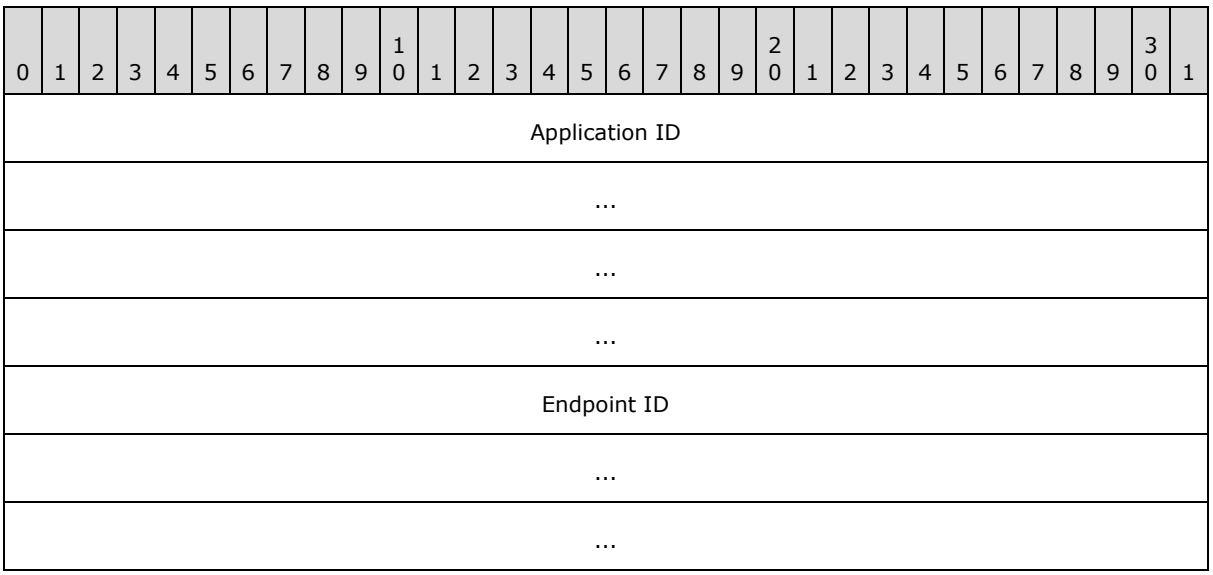
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.25 AddContentItem

This command adds content for the specified **application/endpoint pair**. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000114.

2.2.3.25.1 AddContentItem Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



...
Content ID
Content Data (variable)
...

Application ID (16 bytes): A GUID that identifies the application that is the source of the content.

Endpoint ID (16 bytes): A GUID that identifies the endpoint that is the destination for the content. For a list of platform-defined Endpoint IDs, see the following behavior note.<8>

Content ID (4 bytes): A DWORD that assigns an identifier to the content.

Note Each content item must be associated with a unique ID. Sending new content with a previously sent content ID will result in the new content replacing the previous content.

Content Data (variable): A Counted Byte Array (section 2.2.1.2) that contains the data from the application. The format of the data is specified by the receiving endpoint.

2.2.3.25.2 AddContentItem Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.26 DeleteContentItem

This command removes content for the specified application/endpoint pair. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000115.

2.2.3.26.1 DeleteContentItem Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Application ID																															
...																															
...																															
...																															
Endpoint ID																															

...
...
...
Content ID

Application ID (16 bytes): A GUID that identifies the application that is the source of the content.

Endpoint ID (16 bytes): A GUID that identifies the endpoint that is the destination for the content.

Content ID (4 bytes): A DWORD that identifies the content item to be deleted.

2.2.3.26.2 DeleteContentItem Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.27 DeleteAllContentItems

This command removes all content items for the specified application/endpoint pair. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000116.

2.2.3.27.1 DeleteAllContentItems Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Application ID																															
...																															
...																															
...																															
Endpoint ID																															
...																															
...																															
...																															

Application ID (16 bytes): A GUID that identifies the application that created the content.

Endpoint ID (16 bytes): A GUID that identifies the endpoint that owns the content.

2.2.3.27.2 DeleteAllContentItems Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.28 GetSupportedEndpoints

This command retrieves the list of endpoints the device supports. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000117.

2.2.3.28.1 GetSupportedEndpoints Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000117.

2.2.3.28.2 GetSupportedEndpoints Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Count																															
Endpoints (variable)																															
...																															
...																															
...																															

Count (4 bytes): A DWORD that specifies the number of elements returned in the Endpoints field.

Endpoints (variable): An array of GUIDs that identifies the endpoints available on the device.

2.2.3.29 SetTimeZone

This command specifies the current time zone. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000118.

2.2.3.29.1 SetTimeZone Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Bias																															
Standard Date																															
...																															
...																															
...																															
Standard Bias																															
Daylight Date																															
...																															
...																															
...																															
Daylight Bias																															

Bias (4 bytes): A LONG that specifies the time difference, in minutes, between the time zone's local time and Coordinated Universal Time (UTC).

Standard Date (16 bytes): A SYSTEMTIME structure that contains a date and local time when the transition from daylight saving time to standard time occurs.

Standard Bias (4 bytes): A LONG value, in minutes, to be used during local time translations that occur during standard time.

Daylight Date (16 bytes): A SYSTEMTIME structure that contains a date and local time when the transition from standard time to daylight saving time occurs.

Daylight Bias (4 bytes): A LONG value to be used during local time translations that occur during daylight saving time.

Note See [MS-DTYP] section 2.3.11 for the SYSTEMTIME structure. For information on how the fields of this packet are used see section 9, Appendix D: SetTimeZone Field Usage.

2.2.3.29.2 SetTimeZone Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.30 GetDeviceName

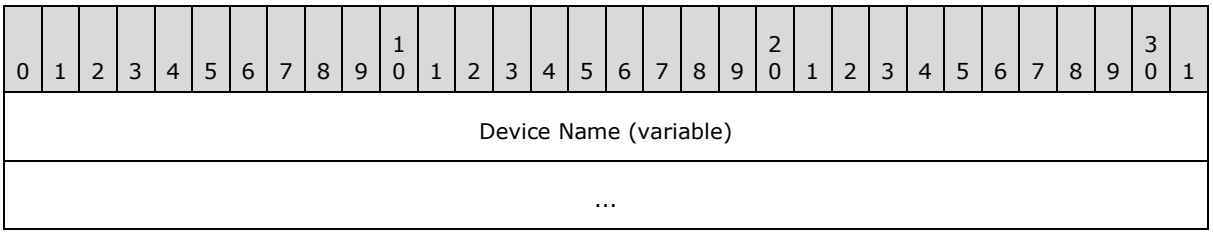
This command retrieves the device name. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000500.

2.2.3.30.1 GetDeviceName Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000500.

2.2.3.30.2 GetDeviceName Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



Device Name (variable): A Counted String (section 2.2.1.1) that specifies the name of the device.

2.2.3.31 GetDeviceManufacturer

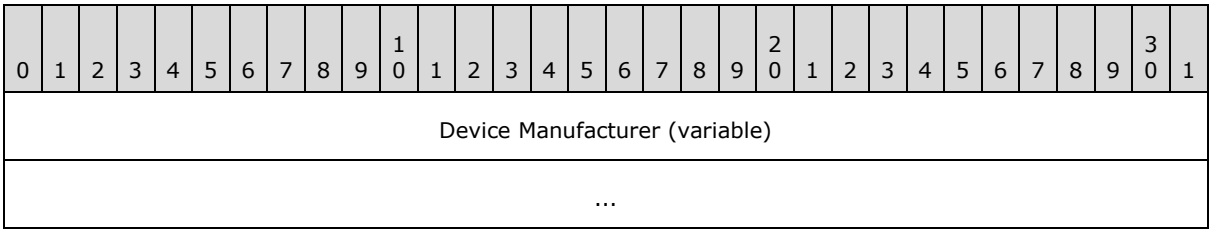
This command retrieves the name of the device manufacturer. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000501.

2.2.3.31.1 GetDeviceManufacturer Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000501.

2.2.3.31.2 GetDeviceManufacturer Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



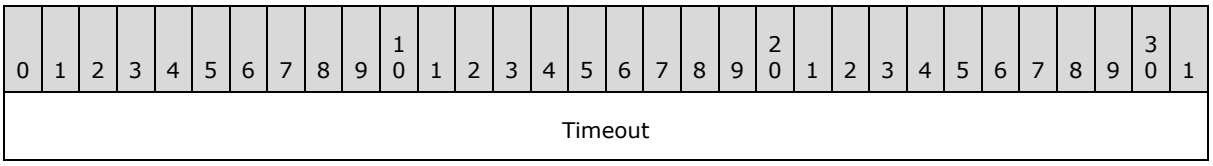
Device Manufacturer (variable): A Counted String (section 2.2.1.1) that specifies the name of the device manufacturer.

2.2.3.32 SetBacklightTimeout

This command sets the timeout, in seconds, before the display panel turns its backlight off due to inactivity. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000503.

2.2.3.32.1 SetBacklightTimeout Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Timeout (4 bytes): A DWORD that specifies the time, in seconds, before the display panel turns its backlight off due to inactivity.

Note A value of zero means that the display backlight does not time out.

2.2.3.32.2 SetBacklightTimeout Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.33 GetBacklightTimeout

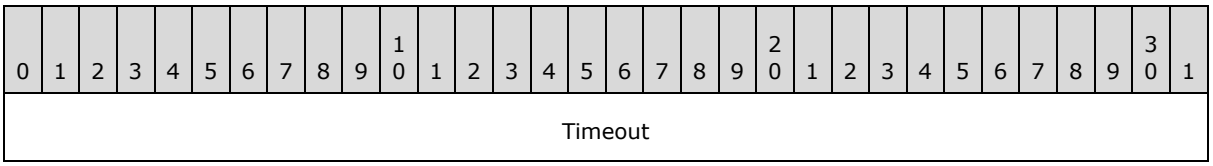
This command reads the timeout, in seconds, before the display panel turns its backlight off due to inactivity. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000504.

2.2.3.33.1 GetBacklightTimeout Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000504.

2.2.3.33.2 GetBacklightTimeout Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



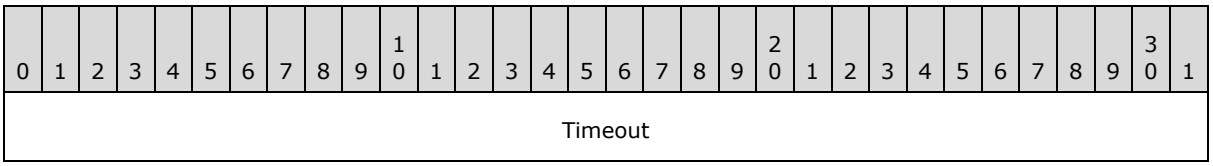
Timeout (4 bytes): A DWORD that specifies the time, in seconds, before the display panel turns its backlight off due to inactivity.

2.2.3.34 SetPanelTimeout

This command sets the timeout, in seconds, before the display panel turns its screen off due to inactivity. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000505.

2.2.3.34.1 SetPanelTimeout Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Timeout (4 bytes): A DWORD that specifies the time, in seconds, before the display powers off due to inactivity.

Note A value of zero disables the display panel's screen-off timeout.

2.2.3.34.2 SetPanelTimeout Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.35 GetPanelTimeout

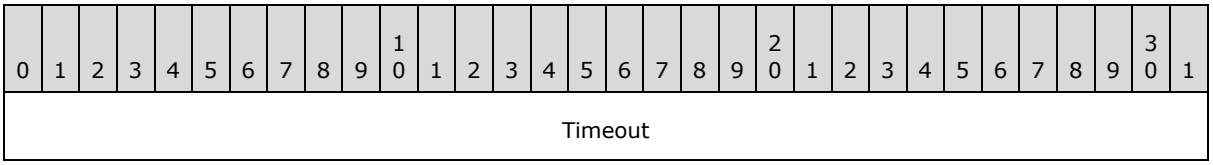
This command reads the timeout, in seconds, before the display panel turns off due to inactivity. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000506.

2.2.3.35.1 GetPanelTimeout Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000506.

2.2.3.35.2 GetPanelTimeout Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



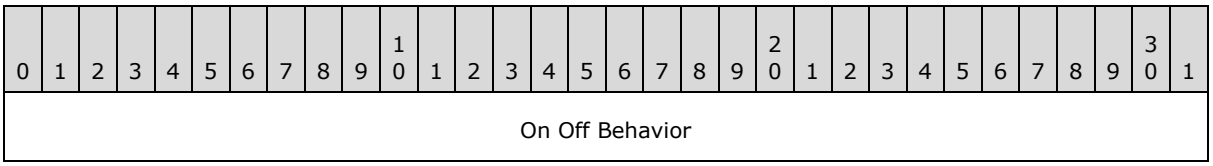
Timeout (4 bytes): A DWORD that specifies the time, in seconds, before the display panel turns off due to inactivity.

2.2.3.36 SetOnOffBehavior

This command indicates to the device whether it should lock out its keypad after the display panel timeout has elapsed. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000509.

2.2.3.36.1 SetOnOffBehavior Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



On Off Behavior (4 bytes): A Boolean value that determines whether the device should lock out its keypad after the display panel timeout has elapsed.

Value	Meaning
0x00000000	Do not lock out the keypad after the display panel timeout has elapsed.
0x00000001	Lock the keypad after the display panel timeout has elapsed.

2.2.3.36.2 SetOnOffBehavior Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.37 GetOnOffBehavior

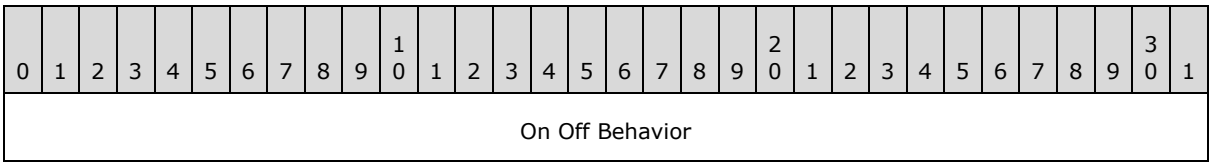
This command reads the state of the device to determine whether it will lock out the keypad after the display panel timeout has elapsed. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00050A.

2.2.3.37.1 GetOnOffBehavior Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x00050A.

2.2.3.37.2 GetOnOffBehavior Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



On Off Behavior (4 bytes): A Boolean value that determines whether the device will lock its keypad after the display panel timeout has elapsed.

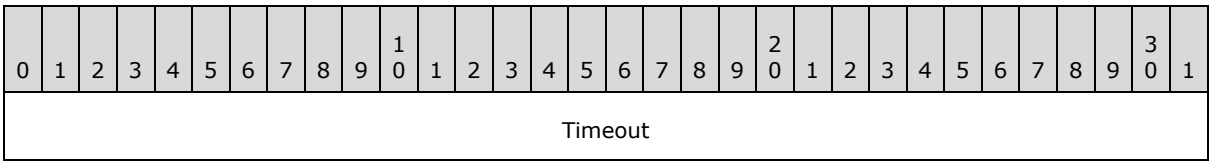
Value	Meaning
0x00000000	Do not lock out the keypad after the display panel timeout has elapsed.
0x00000001	Lock the keypad after the display panel timeout has elapsed.

2.2.3.38 SetLockTimeout

This command sets the timeout, in seconds, before the device PIN-locks itself. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000511.

2.2.3.38.1 SetLockTimeout Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Timeout (4 bytes): A DWORD that specifies the time, in seconds, before the device PIN-locks itself due to inactivity.

Note The following values have special meaning: 0x00000000 disables the timeout; 0x0000FFFF automatically PIN-locks the device when the display is powered down.

2.2.3.38.2 SetLockTimeout Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.39 GetLockTimeout

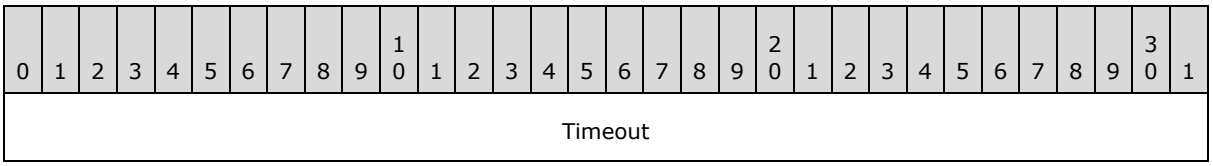
This command reads the time, in seconds, before the device PIN-locks itself due to inactivity. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000512.

2.2.3.39.1 GetLockTimeout Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000512.

2.2.3.39.2 GetLockTimeout Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



Timeout (4 bytes): A DWORD that specifies the time, in seconds, before the device PIN-locks itself due to inactivity.

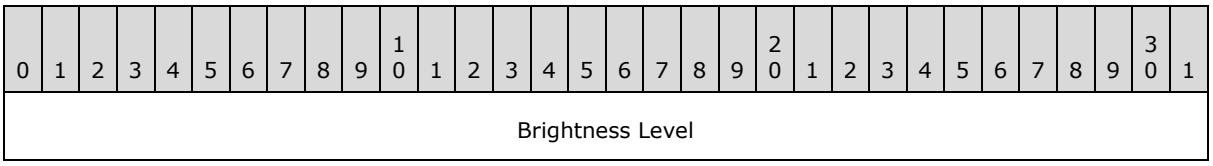
Note The following values have special meaning: 0x00000000 disables the timeout; 0x0000FFFF automatically PIN-locks the device when the display is powered down. When OR'd with the bitmask 0x00010000, the resulting value means that the user has provided a PIN.

2.2.3.40 SetScreenBrightness

This command indicates the brightness level of the display panel. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000514.

2.2.3.40.1 SetScreenBrightness Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Brightness Level (4 bytes): A DWORD that specifies the brightness level of the display.

Value	Brightness Level
0x00000000	off
0x00000001	low
0x00000002	medium
0x00000003	high

2.2.3.40.2 SetScreenBrightness Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.41 GetScreenBrightness

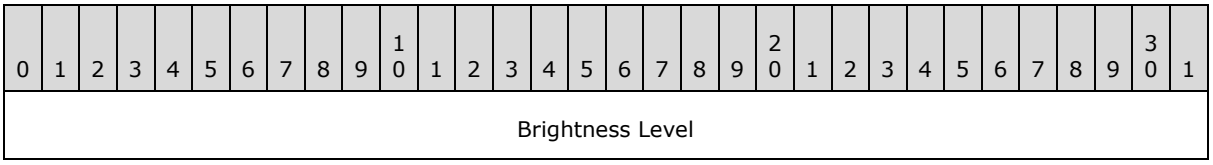
This command reads the brightness level of the display panel. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000513.

2.2.3.41.1 GetScreenBrightness Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000513.

2.2.3.41.2 GetScreenBrightness Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



Brightness Level (4 bytes): A DWORD that specifies the brightness level of the display panel.

Value	Brightness Level
0x00000000	off

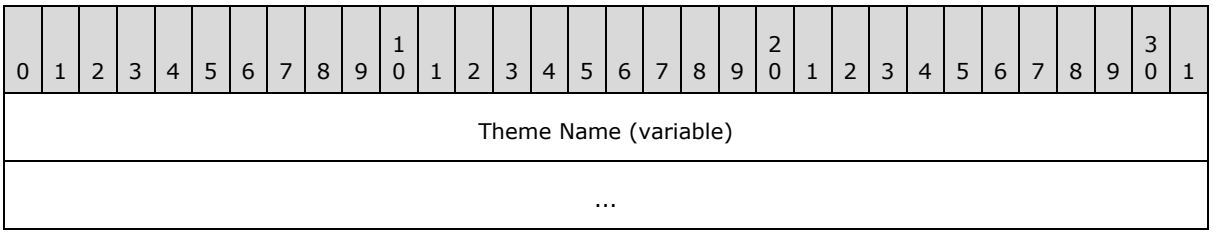
Value	Brightness Level
0x00000001	low
0x00000002	medium
0x00000003	high

2.2.3.42 SetCurrentTheme

This command indicates the name of the current theme for user-interface elements to be set on the device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000515.

2.2.3.42.1 SetCurrentTheme Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Theme Name (variable): A Counted String (section 2.2.1.1) that specifies the name of the theme to use for user interface elements.

2.2.3.42.2 SetCurrentTheme Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.43 GetCurrentTheme

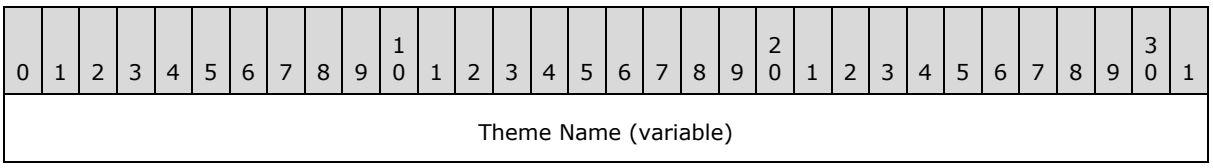
This command reads the name of the current theme for user-interface elements set on the device. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000516.

2.2.3.43.1 GetCurrentTheme Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000516.

2.2.3.43.2 GetCurrentTheme Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



...

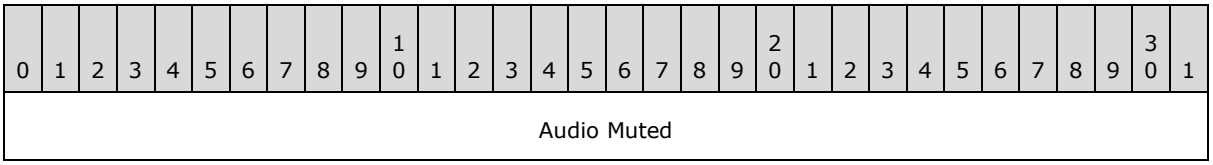
Theme Name (variable): A Counted String (section 2.2.1.1) that specifies the name of the theme to use for user interface elements.

2.2.3.44 SetAudioMuted

This command indicates to the device the audio-muted state of the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000517.

2.2.3.44.1 SetAudioMuted Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Audio Muted (4 bytes): A Boolean value that specifies the audio muted state of the PC host.

Value	Meaning
0x00000000	The PC host's audio is not muted.
0x00000001	The PC host's audio is muted.

2.2.3.44.2 SetAudioMuted Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.45 GetAudioMuted

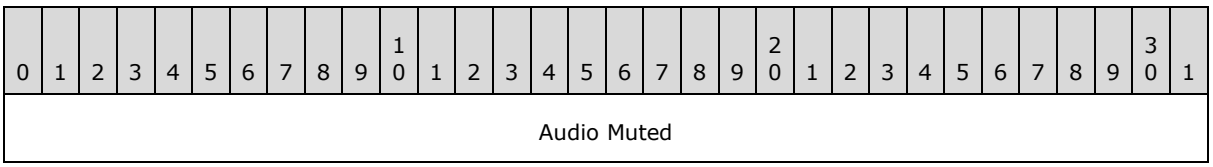
This command retrieves the audio-muted state of the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000518.

2.2.3.45.1 GetAudioMuted Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000518.

2.2.3.45.2 GetAudioMuted Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



Audio Muted (4 bytes): A Boolean value that specifies the audio muted state of the PC host.

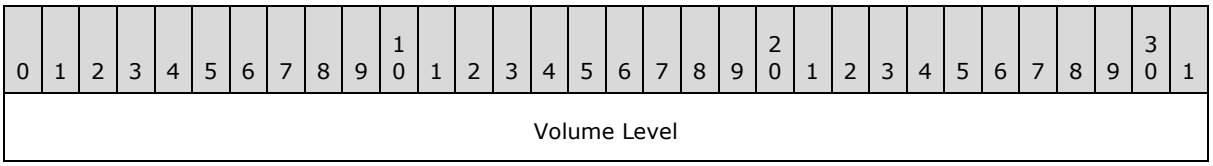
Value	Meaning
0x00000000	The PC host's audio is not muted.
0x00000001	The PC host's audio is muted.

2.2.3.46 SetAudioVolume

This command informs the device about the current volume level of the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000519.

2.2.3.46.1 SetAudioVolume Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Volume Level (4 bytes): A DWORD value that indicates the current value of the PC host's audio volume. The volume level ranges from 0x00000000 to 0x0000FFFF, where 0x00000000 is the lowest volume level.

2.2.3.46.2 SetAudioVolume Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.47 GetAudioVolume

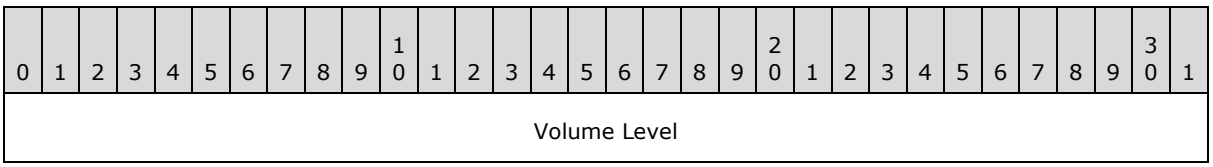
This command reads the current volume level of the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00051A.

2.2.3.47.1 GetAudioVolume Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x00051A.

2.2.3.47.2 GetAudioVolume Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



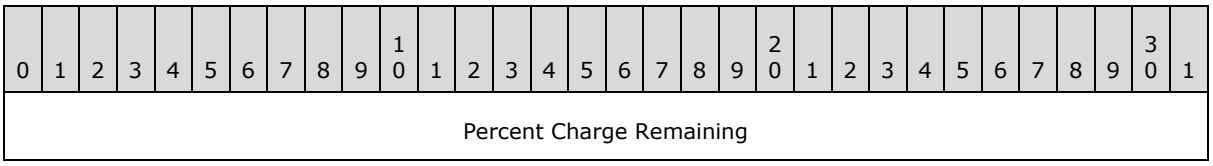
Volume Level (4 bytes): A DWORD that indicates the current value of the PC host's audio volume. The volume level ranges from 0x00000000 to 0x0000FFFF, where 0x00000000 is the lowest volume level.

2.2.3.48 SetBatteryRemainingCapacity

This command informs the device about the percentage of full battery charge that remains on the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00051D.

2.2.3.48.1 SetBatteryRemainingCapacity Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Percent Charge Remaining (4 bytes): A DWORD that indicates the percentage of full battery charge remaining. This field can be a value in the range of 0x00000000 to 0x00000064; if the status is unknown, the value is 0x000000FF.

2.2.3.48.2 SetBatteryRemainingCapacity Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.49 GetBatteryRemainingCapacity

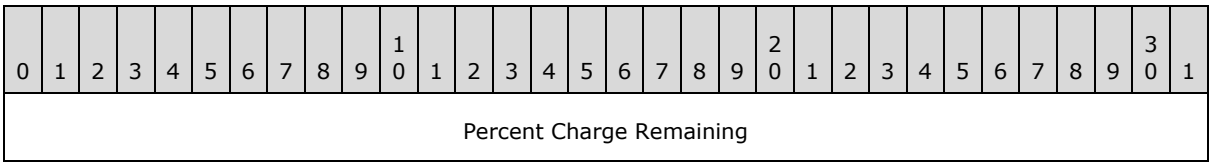
This command reads the percentage of full battery charge that remains on the host PC. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00051E.

2.2.3.49.1 GetBatteryRemainingCapacity Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x00051E.

2.2.3.49.2 GetBatteryRemainingCapacity Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.

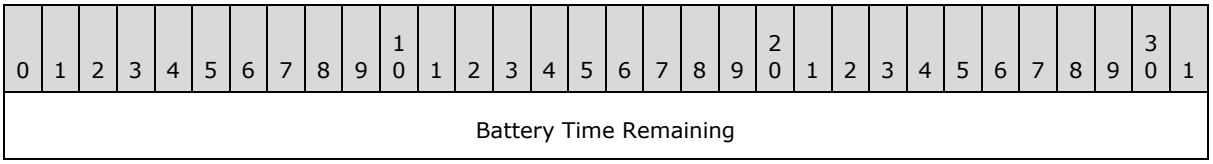


Percent Charge Remaining (4 bytes): A DWORD that indicates the percentage of full battery charge remaining. This field can be a value in the range of 0x00000000 to 0x00000064; if the status is unknown, the value is 0x000000FF.

2.2.3.50 SetBatteryTimeToDischarge

This command informs the device about the number of seconds of battery life remaining for the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00051F.

2.2.3.50.1 SetBatteryTimeToDischarge Command



Battery Time Remaining (4 bytes): A LONG value that indicates the number of seconds of battery life remaining for the PC host. The value is 0xFFFFFFFF if the remaining seconds are unknown.

2.2.3.50.2 SetBatteryTimeToDischarge Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.51 GetBatteryTimeToDischarge

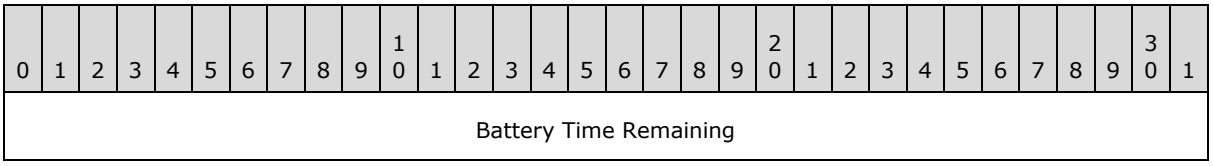
This command reads the number of seconds of battery life remaining for the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000520.

2.2.3.51.1 GetBatteryTimeToDischarge Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000520.

2.2.3.51.2 GetBatteryTimeToDischarge Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



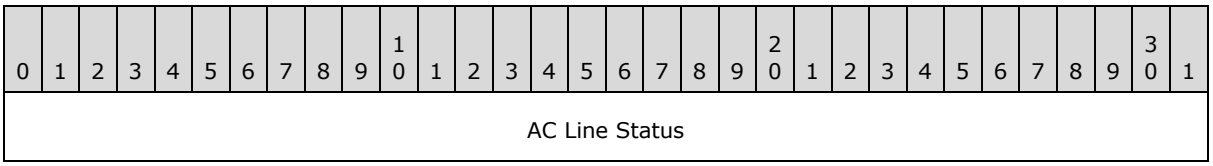
Battery Time Remaining (4 bytes): A LONG value that indicates the number of seconds of battery life remaining for the PC host. The value is 0xFFFFFFFF if the remaining seconds are unknown.

2.2.3.52 SetBatteryAcLineStatus

This command indicates to the device the AC line status of the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000521.

2.2.3.52.1 SetBatteryAcLineStatus Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



AC Line Status (4 bytes): A DWORD that indicates whether the PC host is plugged into the AC power.

Value	AC Line Status
0x00000000	Disconnected
0x00000001	Connected
0x000000FF	Unknown

2.2.3.52.2 SetBatteryAcLineStatus Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.53 GetBatteryAcLineStatus

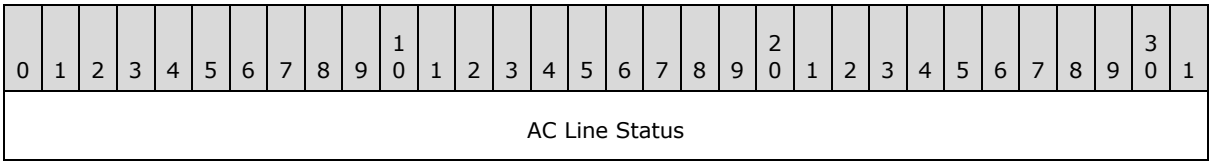
This command reads the status of AC line of the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000522.

2.2.3.53.1 GetBatteryAcLineStatus Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000522.

2.2.3.53.2 GetBatteryAcLineStatus Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



AC Line Status (4 bytes): A DWORD that indicates whether the PC host is plugged into the AC power.

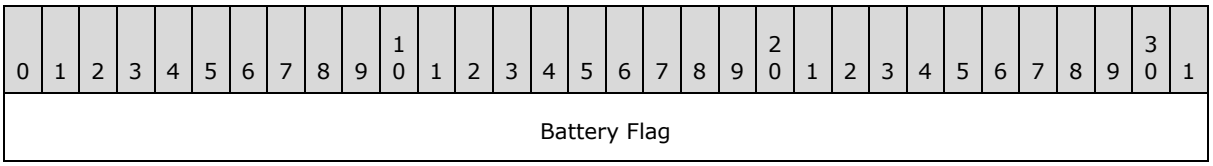
Value	AC Line Status
0x00000000	Disconnected
0x00000001	Connected
0x000000FF	Unknown

2.2.3.54 SetBatteryFlag

This command informs the AXDS device about the battery charge status of the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000523.

2.2.3.54.1 SetBatteryFlag Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Battery Flag (4 bytes): A DWORD that indicates the battery charge status of the PC host.

Value	Meaning
0x00000001	High - The battery capacity is at more than 66 percent.
0x00000002	Low - The battery capacity is at less than 33 percent.
0x00000004	Critical - The battery capacity is near depletion.
0x00000008	Charging.
0x00000080	No system battery.
0x000000FF	Unknown status - Unable to read the battery flag information.

2.2.3.54.2 SetBatteryFlag Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.55 GetBatteryFlag

This command reads the battery charge status of the PC host. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000524.

2.2.3.55.1 GetBatteryFlag Command

The AXDS packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000524.

2.2.3.55.2 GetBatteryFlag Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Battery Flag																															

Battery Flag (4 bytes): A DWORD that indicates the battery charge status of the PC host.

Value	Meaning
0x00000001	High - The battery capacity is at more than 66 percent.
0x00000002	Low - The battery capacity is at less than 33 percent.
0x00000004	Critical - The battery capacity is near depletion.
0x00000008	Charging.
0x00000080	No system battery.
0x000000FF	Unknown status - Unable to read the battery flag information.

2.2.3.56 SetWirelessNetworks

This command informs the device about the list of wireless networks available to the PC host and status information for each network. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000525.

2.2.3.56.1 SetWirelessNetworks Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Network Type (variable)																															
...																															
Network Count																															
Network Data (variable)																															
...																															

Network Type (variable): A Counted String (section 2.2.1.1) that specifies the type of network; for example, "WiFi".

Network Count (4 bytes): A DWORD that indicates the number of discovered networks.

Network Data (variable): An array of Network Data structures that represent the discovered networks. Each element in the array has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Friendly Name (variable)																															
...																															
Status																															
Signal Strength																															

Friendly Name (variable): A Counted String (section 2.2.1.1) that specifies the friendly name (SSID) of the network.

Status (4 bytes): A DWORD that indicates the connection status of the network.

Value	Meaning
0x00000000	Not connected
0x00000001	Connected

Signal Strength (4 bytes): A LONG that represents the received signal strength of the network. This field contains a value between 0 and 100.

2.2.3.56.2 SetWirelessNetworks Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.57 SetWirelessCapable

This command indicates to the device whether the PC host is capable of connecting to a specified wireless network type. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000526.

2.2.3.57.1 SetWirelessCapable Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Network Type (variable)																															
...																															
Capability																															

Network Type (variable): A Counted String (section 2.2.1.1) that specifies the type of network; for example, "WiFi".

Capability (4 bytes): A Boolean value that indicates the capability to connect to the type of wireless network.

Value	Meaning
0x00000000	Not capable of connecting to the specified wireless network type.
0x00000001	Capable of connecting to the specified wireless network type.

2.2.3.57.2 SetWirelessCapable Response

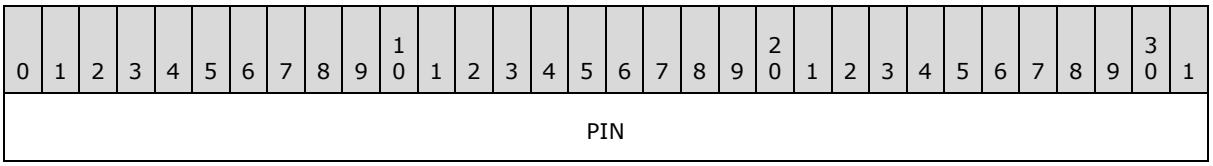
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.58 ResetPin

This command clears any previous PIN associated with the device, and sets it to the new value. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000528.

2.2.3.58.1 ResetPin Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



PIN (4 bytes): A DWORD that specifies a 4-digit PIN used to lock the device. The PIN can be a value in the range of 0x00000000-0x0000270F (or 0-9999 in decimal). Any value outside this range (as in >9999 in decimal) indicates that there is no PIN.

2.2.3.58.2 ResetPin Response

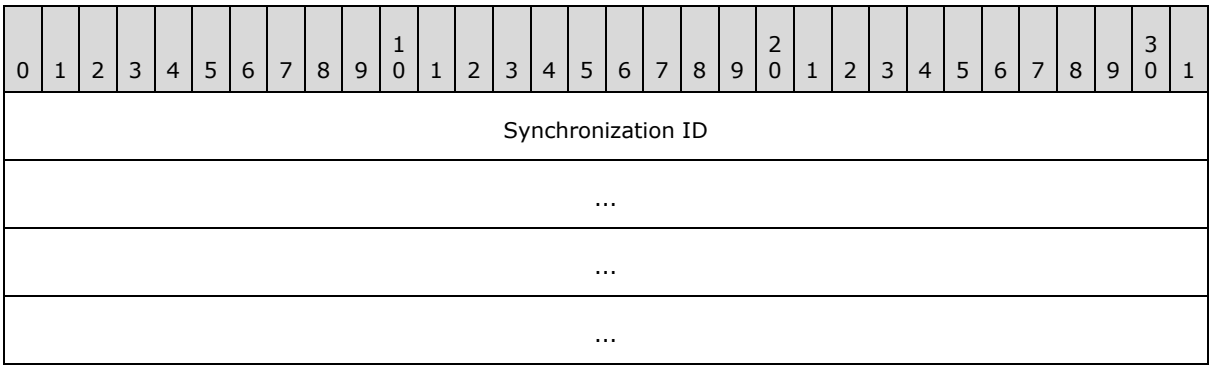
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.3.59 Sync

This command denotes a Sync packet, which is the first packet that a device receives each time the transport connection is opened. The Sync packet notifies a device that a protocol session has begun. When a device receives this command packet, the device must verify that the GUID corresponds to the protocol version that the device supports. If the device supports the protocol, the device must clear any data that is stored in the receive and send buffers, and then send a response packet that contains the same GUID. Both the PC host and the device then reset their respective packet sequence numbers to one. If the device does not support the protocol, the device must reply with a NAK packet. Sync packets always have a packet sequence number of zero. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000502.

2.2.3.59.1 Sync Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.

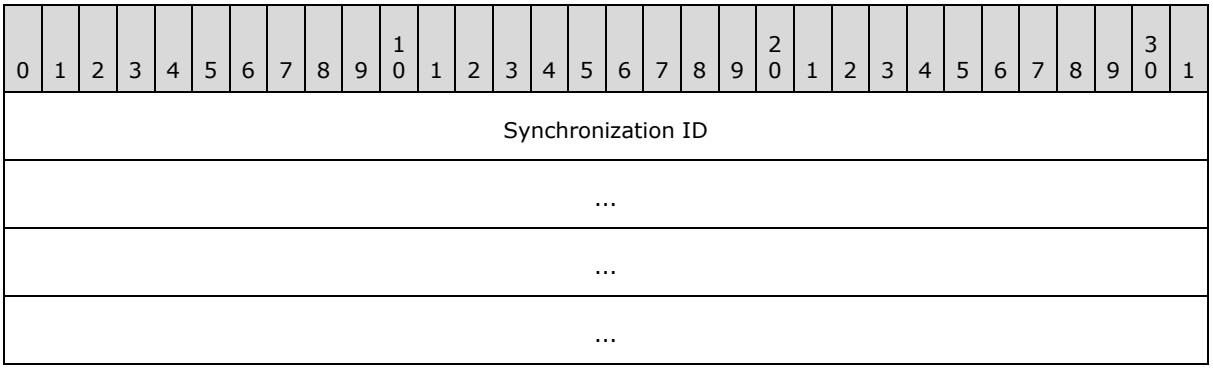


Synchronization ID (16 bytes): A GUID that specifies the synchronization ID. The following table shows the defined synchronization IDs.

Value	Description
a33f248b-882f-4531-82c2-ed3b90c5c520	Defines the standard set of protocol packets.
77af0703-d1b9-4fc7-b40e-08bfb7e14cc9	Defines the extended set of protocol packets.

2.2.3.59.2 Sync Response

Following the AXDS packet header, the response packet must contain the data in the following format, in the order shown below.



Synchronization ID (16 bytes): A GUID that specifies the synchronization ID. The following table shows the defined synchronization IDs.

Value	Description
a33f248b-882f-4531-82c2-ed3b90c5c520	Defines the standard set of protocol packets.
77af0703-d1b9-4fc7-b40e-08bfb7e14cc9	Defines the extended set of protocol packets.

The classification of the standard and extended set of protocol packets is described in section 7, Appendix B: Packet ID Description Table.

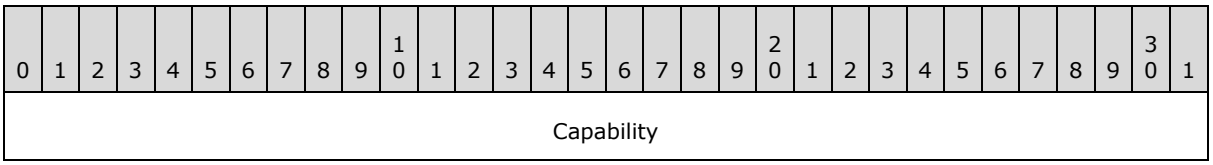
The extended set includes the standard set.

2.2.3.60 SetAudioCapable

This command indicates whether the PC host has audio capability. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x00051B.

2.2.3.60.1 SetAudioCapable Command

Following the AXDS packet header, the command packet must contain the data in the following format, in the order shown below.



Capability (4 bytes): A Boolean value that indicates the availability of audio capability on the PC host.

Value	Meaning
0x00000000	Does not have audio capability.
0x00000001	Has audio capability.

2.2.3.60.2 SetAudioCapable Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.4 Event Packets

Events from an AXDS device to the AXDS platform are the primary means of communication between a device and an application that runs on the computer. As such, it is very important that devices and drivers properly implement events in order to ensure proper application functionality. Events can be sent across the wire asynchronously to command/response packets. On the AXDS device side, there are three main types of events that can be generated and posted to the platform from the device driver: application events, content missing events, and user change events.

A user change event should be sent by the host PC to request a change in the current user of the AXDS device (in other words, which user's data is sent to the device). This type of event is applicable only to AXDS devices that operate in the "assigned" user mode. A content missing event is sent by a device when a specific piece of content is required but not currently available to the device and the content is not in the device's local cache. Typically, when content is not available, a content missing event is sent on demand to the device when the content is referenced or displayed. The application event is a generic event mechanism whose specific content and interpretations depend on the endpoint from which the event is sent.

Event packets represent events that the AXDS device needs the AXDS platform to be aware of. The following sections describe event packet types. Event packets follow the same command/response structure as defined in section 2.2.2, Auxiliary Display Protocol Packet Header.

2.2.4.1 Ping

This event provides a way for the AXDS device to test a connection with the host PC. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x000001.

2.2.4.1.1 Ping Event

The Auxiliary Display Protocol (AXDS) packet header with the Packet Type ID field (section 2.2.2.2) set to 0x000001.

2.2.4.1.2 Ping Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.4.2 ContentMissing

This event provides notification about missing content for the specified application/endpoint pair. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x004000.

2.2.4.2.1 ContentMissing Event

Following the AXDS packet header, the event packet must contain the data in the following format, in the order shown below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Application ID																															
...																															
...																															
...																															
Endpoint ID																															
...																															
...																															
...																															
Content ID																															

Application ID (16 bytes): A GUID that identifies the application that is the source of the content.

Endpoint ID (16 bytes): A GUID that identifies the endpoint that is requesting the content.

Content ID (4 bytes): A DWORD that identifies the requested content item.

2.2.4.2.2 ContentMissing Response

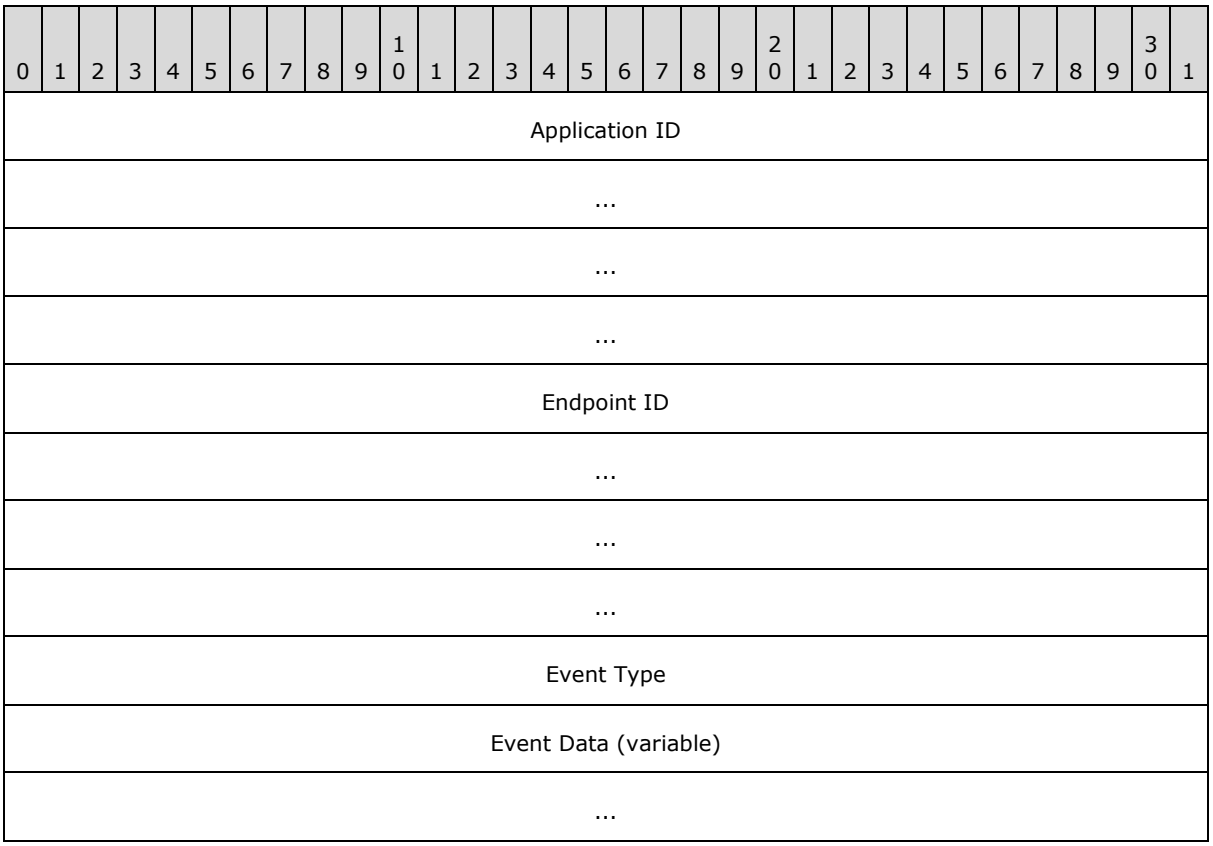
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.4.3 ApplicationEvent

This event provides an application-defined event notification. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x004001.

2.2.4.3.1 ApplicationEvent Event

Following the AXDS packet header, the event packet must contain the data in the following format, in the order shown below.



Application ID (16 bytes): A GUID that identifies the application that is the destination for the event.

Endpoint ID (16 bytes): A GUID that identifies the endpoint that is the source of the event. For a list of platform-defined Endpoint IDs, see the following behavior note.<9>

Event Type (4 bytes): A DWORD that specifies the type of event data being sent. The Event Type values are defined by the endpoint.

Event Data (variable): A Counted Byte Array (section 2.2.1.2) that contains the event data from the endpoint. The format of the data is specified by the endpoint.

2.2.4.3.2 ApplicationEvent Response

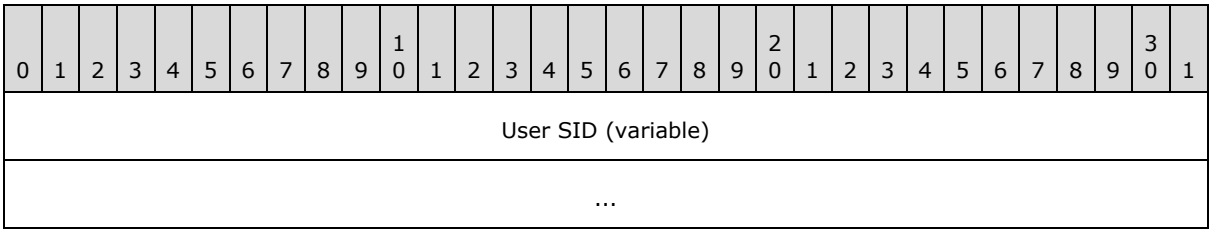
The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

2.2.4.4 ChangeUserRequestEvent

This event requests that a new, available user take ownership of the device. The new user must be identified by a SID. The Packet Type ID field (section 2.2.2.2) in the packet header is set to 0x004002.

2.2.4.4.1 ChangeUserRequestEvent Event

Following the AXDS packet header, the event packet must contain the data in the following format, in the order shown below.



User SID (variable): A Counted String (section 2.2.1.1) that contains the SID of the user that should take ownership of the device.

2.2.4.4.2 ChangeUserRequestEvent Response

The response is simply an ACK or a NAK in the Control Data field (section 2.2.2.1) of the AXDS packet header.

3 Protocol Details

3.1 Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

See section 2.2 for a detailed explanation about AXDS message processing rules.

3.1.5.1 Processing Overview

Either the host PC or the AXDS-compatible device can start communications. An AXDS host can send commands to an AXDS-compatible device in the form of a packet. The device returns a packet that indicates the success or failure of the command. Additionally, the device can send event information to the host. These events can provide notifications such as cache-content-missing notifications, application events, and user change events. The host acknowledges receipt of the event from the device by responding with a packet.

String fields within packets are composed of a 4-byte value that indicates the length (in characters) of the string, followed by the string data. String data consists of 16-bit Unicode characters stored in big-endian byte order (bytes are numbered from left to right). Strings must not be null terminated. Buffer fields within packets are formatted, as with strings, as a 4-byte value that indicates the length, followed by the buffer data. String or buffer fields that contain no data are formatted as a 4-byte value that indicates zero-length.

The bytes of a numeric value are always transmitted in little-endian order.

3.1.5.2 Command Packets

Command packets represent the operations that the AXDS platform can request the device to perform. The host always initiates command packets. Command packets can be composed of several pieces of data that constitute arguments for the command. For every command, the device must return a response packet.

3.1.5.3 Response Packets

Response packets must be sent as a reply to command packets and event packets. A well-formed response packet header must have the following format:

- The packet number must be set to the same packet number as the command packet that is being responded to. This setting ensures that the response can be matched correctly with the command.
- The packet type must be set to the same packet type as the command packet that is being responded to. The packet type's high bit must be set to indicate that the packet is a response packet.

The response packet type can be either ACK or NAK, as described in the following sections.

3.1.5.3.1 ACK

For a positive response packet, specify ACK by clearing bit 6 of the high byte of the packet-type part of the packet header. Some ACK response packets may also contain data that the command has requested. The descriptions for the command packets include the specifics of the data that is sent in response to a particular command.

3.1.5.3.2 NAK

For a negative response packet, specify NAK by setting bit 6 of the high byte of the packet-type part of the packet header. In addition, bits 5 through 0 can contain an error code for the failure of the command that is being responded to.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

4.1 A Command with Data, a Response without Data

The following is an example of a command that contains data, along with the corresponding acknowledgment.

```
[Send] - 07-15-09 05:25 PM

        [AddContentItem] -- 0x000114
        SEQ: 158  :     ACK ECode:0x00 size: 98

  169616: 62 00 00 00 14 01 00 00 9e 00 97 ec 4d 40 5a 07  : b...........M@Z.
  169632: c5 4b a1 b6 d0 58 70 48 4b 38 3f 35 a5 a9 4b 2d  : .K...XpHK8?5..K-
  169648: ce 47 93 ee 75 9f 3a 7d da 4f 00 00 00 00 30 00  : .G..u.:}.O....0.
  169664: 00 00 34 25 20 43 50 55 20 2d 20 32 36 25 20 4d  : ..4% CPU - 26% M
  169680: 65 6d 6f 72 79 0d 0a 74 65 63 72 61 73 20 3a 20  : emory..tecras :
  169696: 4c 6f 63 61 6c 53 74 61 6e 64 61 72 64 55 73 65  : LocalStandardUse
  169712: 72 00                                            : r.


----------------------------------------

[Read] - 07-15-09 05:25 PM

        [AddContentItem] -- 0x000114
        SEQ: 158  :RESP ACK ECode:0x00 size: 10

  02704: 0a 00 00 00 14 01 00 80 9e 00                    : ..........


----------------------------------------
```

4.2 A Command without Data, a Response with Data

The following is an example of a command that does not contain data, along with the response that does contain data.

```
----------------------------------------

[Send] - 07-16-09 06:44 PM

        [GetCurrentTheme] -- 0x000516
        SEQ: 105  :     ACK ECode:0x00 size: 10

  90032: 0a 00 00 00 16 05 00 00 69 00                    : ........i.


----------------------------------------

[Read] - 07-16-09 06:44 PM

        [GetCurrentTheme] -- 0x000516
        SEQ: 105  :RESP ACK ECode:0x00 size: 28
```

```
  02272: 1c 00 00 00 16 05 00 80 69 00 07 00 00 00 57 00  : ........i.....W.
  02288: 41 00 56 00 45 00 31 00 30 00 30 00              : A.V.E.1.0.0.


---------------------------------------
```

4.3 An Event with Data

The following is an example of an event that contains data, along with the corresponding acknowledgment.

```
[Read] - 07-15-09 03:21 PM

        [ApplicationEvent] -- 0x004001
        SEQ: 1    :     ACK ECode:0x00 size: 50

  05904: 32 00 00 00 01 40 00 00 01 00 3f 3a 1d ba 69 56  : 2....@....?:..iV
  05920: cc 48 95 7b 00 a9 7c f0 46 fb 3f 35 a5 a9 4b 2d  : .H.{..|.F.?5..K-
  05936: ce 47 93 ee 75 9f 3a 7d da 4f 00 00 ff ff 00 00  : .G..u.:}.O......
  05952: 00 00                                            : ..


---------------------------------------

[Send] - 07-15-09 03:21 PM

        [ApplicationEvent] -- 0x004001
        SEQ: 1    :RESP ACK ECode:0x00 size: 10

  52480: 0a 00 00 00 01 40 00 80 01 00                    : .....@....


---------------------------------------
NOTE Here is a device operation that requests content from the gadget running on the PC host.
Therefore [Send] is associated with messages sent from the PC host to the device and [Read]
is associated with messages sent from the device to the PC host.
```

5 Security

5.1 Security Considerations for Implementers

AXDS does not provide secure communications. If communication security is required, it must be provided at the transport level.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 1.3: AXDS is not included in any shipped version of the Windows operating system, but is available through the Windows Update mechanism.

<2> Section 2.2.3.4.2: For information about user models, see User Models [MSDN-WDKUM] in the Windows SideShow section of the Windows Driver Kit.

<3> Section 2.2.3.5.2: For information about user models, see User Models [MSDN-WDKUM] in the Windows SideShow section of the Windows Driver Kit.

<4> Section 2.2.3.6: For information about user models, see User Models [MSDN-WDKUM] in the Windows SideShow section of the Windows Driver Kit.

<5> Section 2.2.3.8.1: The Property Keys comprise a GUID and a PID value. A GUID and the following 4 bytes will have a meaning corresponding to a capability, such as screen width. The type can be either a string or a DWORD. For more information, see [MSDN-DEVCAP].

AXDS-compatible devices must provide accurate responses for all standard platform capabilities. The driver can also support additional capabilities, as required. However, supported additional capabilities must always be acknowledged by using an ACK with the VT_EMPTY type, or NAK otherwise.

<6> Section 2.2.3.8.2: These types are defined in the WindowsSideShow.h header, as shown in the following table.

Table 1

Capability	Type	Value
SIDESHOW_CAPABILITY_DEVICE_ID	VT_LPWSTR	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 1
SIDESHOW_CAPABILITY_SCREEN_TYPE	VT_I4	0x8abc88a8, 0x857b,

Capability	Type	Value
		0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 2
SIDESHOW_CAPABILITY_SCREEN_WIDTH	VT_UI2	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 3
SIDESHOW_CAPABILITY_SCREEN_HEIGHT	VT_UI2	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 4
SIDESHOW_CAPABILITY_COLOR_DEPTH	VT_UI2	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 5
SIDESHOW_CAPABILITY_COLOR_TYPE	VT_I4	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 6
SIDESHOW_CAPABILITY_DATA_CACHE	VT_BOOL	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 7
SIDESHOW_CAPABILITY_SUPPORTED_LANGUAGES	VT_LPWSTR	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 8
SIDESHOW_CAPABILITY_CURRENT_LANGUAGE	VT_LPWSTR	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 9
SIDESHOW_CAPABILITY_SUPPORTED_THEMES	VT_LPWSTR	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 10
SIDESHOW_CAPABILITY_SUPPORTED_IMAGE_FORMATS	VT_LPWSTR	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 14
SIDESHOW_CAPABILITY_CLIENT_AREA_WIDTH	VT_UI2	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 15
SIDESHOW_CAPABILITY_CLIENT_AREA_HEIGHT	VT_UI2	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 16

Capability	Type	Value
SIDESHOW_CAPABILITY_DEVICE_ICON	VT_UI1 | VT_VECTOR	0x8abc88a8, 0x857b, 0x4ad7, 0xa3, 0x5a, 0xb5, 0x94, 0x2f, 0x49, 0x2b, 0x99, 17

<7> Section 2.2.3.18.1: The following table lists possible values for platform-defined Endpoint IDs. The values are GUIDs.

Table 2

Value	Meaning
{A9A5353F-2D4B-47ce-93EE-759F3A7DDA4F}	Simple Content Format (SCF)
{4DFF36B5-9DDE-4F76-9A2A-96435047063D}	iCalendar format

Note Custom endpoint values are also permissible.

<8> Section 2.2.3.25.1: Table 2 shown above lists possible values for platform-defined Endpoint IDs. The values are GUIDs.

Note Custom endpoint values are also permissible.

<9> Section 2.2.4.3.1: Table 2 shown above lists possible values for platform-defined Endpoint IDs. The values are GUIDs.

Note Custom endpoint values are also permissible.

7 Appendix B: Packet ID Description Table

Pkt.No.	Packet Type	PacketID (Hex)	Classification	Description
1	Ping	0x1	standard	Provides a way for the host PC to test a connection with the AXDS device.
2	SendPassThrough	0x2	standard	Provides an extensibility mechanism that enables device manufacturers to define methods for sending custom, device-specific commands to a device.
3	Reset	0x3	standard	Restarts the device.
4	SetUserState	0x50	standard	Notifies user-associated devices as to which users are available to be selected as the owner of the device.
5	SetCurrentUser	0x100	standard	Sets the current user of the device.
6	GetCurrentUser	0x101	standard	Retrieves the SID in string format for the current user of the device.
7	GetDeviceFirmwareVersion	0x102	standard	Retrieves a string that identifies the version of the device firmware.
8	GetCapabilities	0x103	standard	Retrieves a device-capability value.
9	GetApplicationOrder	0x104	standard	Retrieves a display-ordered list of AXDS applications that have been added to the device.
10	SetApplicationOrder	0x105	standard	Specifies the display order for AXDS applications that are running on the device.
11	SetLanguage	0x106	standard	Specifies the current language and font size for a device.
12	GetPreEnabledApplications	0x107	standard	Retrieves a list of applications, specified by the device, to be enabled by default for all users of the computer.
13	SetTime	0x108	standard	Specifies the current Coordinated Universal Time (UTC).
14	SetShortDateFormat	0x109	standard	Specifies the format for

Pkt.No.	Packet Type	PacketID (Hex)	Classification	Description
				displaying a concise date.
15	SetLongDateFormat	0x10A	standard	Indicates the format for displaying a full date.
16	SetShortTimeFormat	0x10B	standard	Specifies the format for displaying a concise time.
17	SetLongTimeFormat	0x10C	standard	Indicates the format for displaying a full time.
18	AddApplication	0x10D	standard	Adds a new application to the AXDS device.
19	DeleteApplication	0x10E	standard	Removes a previously added application from the device.
20	DeleteAllApplications	0x10F	standard	Removes all previously added applications from the device, the content for the applications, and their notifications.
21	AddNotification	0x110	standard	Adds a notification that is associated with the specified application.
22	DeleteNotification	0x111	standard	Removes a notification from the specified application.
23	DeleteAllNotifications	0x112	standard	Removes all notifications from the specified application.
24	SetNotificationsEnabled	0x113	standard	Specifies whether notifications are enabled or not.
25	AddContentItem	0x114	standard	Adds content for the specified application/endpoint pair.
26	DeleteContentItem	0x115	standard	Removes content for the specified application/endpoint pair.
27	DeleteAllContentItems	0x116	standard	Removes all content items for the specified application/endpoint pair.
28	GetSupportedEndpoints	0x117	standard	Retrieves the list of endpoints the device supports.
29	SetTimeZone	0x118	standard	Specifies the current time zone.
30	GetDeviceName	0x500	standard	Retrieves the device name.
31	GetDeviceManufacturer	0x501	standard	Retrieves the name of the device manufacturer.

Pkt.No.	Packet Type	PacketID (Hex)	Classification	Description
32	Sync	0x502	standard	Denotes a Sync packet which is the first packet that a device receives each time the transport connection is opened.
33	SetBacklightTimeout	0x503	extended	Sets the timeout, in seconds, before the display panel turns its backlight off due to inactivity.
34	GetBacklightTimeout	0x504	extended	Reads the timeout, in seconds, before the display panel turns its backlight off due to inactivity.
35	SetPanelTimeout	0x505	extended	Sets the timeout, in seconds, before the display panel turns its screen off due to inactivity.
36	GetPanelTimeout	0x506	extended	Reads the timeout, in seconds, before the display panel turns off due to inactivity.
37	SetOnOffBehavior	0x509	extended	Indicates to the device whether it should lock out its keypad after the display panel timeout has elapsed.
38	GetOnOffBehavior	0x50A	extended	Reads the state of the device to see whether it should lock out the keypad after the display panel timeout has elapsed.
39	SetLockTimeout	0x511	extended	Sets the timeout in seconds before the device PIN-locks itself.
40	GetLockTimeout	0x512	extended	Reads the time, in seconds, before the device PIN-locks itself due to inactivity.
41	SetScreenBrightness	0x514	extended	Indicates the brightness level of the display panel.
42	GetScreenBrightness	0x513	extended	Reads the brightness level of the display panel.
43	SetCurrentTheme	0x515	extended	Indicates the name of the current theme for user-interface elements to be set on the device.
44	GetCurrentTheme	0x516	extended	Reads the name of the current theme for user-interface

Pkt.No.	Packet Type	PacketID (Hex)	Classification	Description
				elements set on the device.
45	SetAudioMuted	0x517	extended	Indicates to the device the audio-muted state of the PC host.
46	GetAudioMuted	0x518	extended	Retrieves the audio-muted state of the PC host.
47	SetAudioVolume	0x519	extended	Informs the device of the current volume level of the PC host.
48	GetAudioVolume	0x51A	extended	Reads the current volume level of the PC host.
49	SetAudioCapable	0x51B	extended	Indicates if the PC host has audio capability.
50	SetBatteryRemainingCapacity	0x51D	extended	Informs the device of the percentage value of the PC host's full battery charge remaining.
51	GetBatteryRemainingCapacity	0x51E	extended	Reads the percentage of full battery charge remaining on the host PC.
52	SetBatteryTimeToDischarge	0x51F	extended	Informs the device of the number of seconds of battery life remaining of the PC host.
53	GetBatteryTimeToDischarge	0x520	extended	Reads battery time remaining.
54	SetBatteryAcLineStatus	0x521	extended	Indicates to the device the AC line status of the PC host.
55	GetBatteryAcLineStatus	0x522	extended	Reads the status of the AC line of the PC host.
56	SetBatteryFlag	0x523	extended	Informs the AXDS device of the battery charge status of the PC host.
57	GetBatteryFlag	0x524	extended	Reads the battery charge status of the PC host.
58	SetWirelessNetworks	0x525	extended	Informs the device of a list of wireless networks available to the PC host and the status information for each network.
59	SetWirelessCapable	0x526	extended	Indicates to the device whether the PC host is capable of connecting to a specified wireless network type or not.

Pkt.No.	Packet Type	PacketID (Hex)	Classification	Description
60	ResetPin	0x528	extended	Clears any previous PIN associated with the device, and sets it to the new value.
61	ContentMissing	0x4000	standard	Provides notification about missing content for the specified application/endpoint pair.
62	ApplicationEvent	0x4001	standard	Provides an application-defined event notification.
63	ChangeUserRequestEvent	0x4002	standard	Requests that a new, available user take ownership of the device.

8 Appendix C: Date and Time Formats

A date or time format is specified by a pattern string that consists of a concatenation of one or more format specifiers [MSDN-Date-Format] [MSDN-Time-Format].

Format specifier	Description	Examples
"d"	The day of the month, from 1 through 31.	6/1/2009 1:45:30 PM -> 1 6/15/2009 1:45:30 PM -> 15
"dd"	The day of the month, from 01 through 31.	6/1/2009 1:45:30 PM -> 01 6/15/2009 1:45:30 PM -> 15
"ddd"	The abbreviated name of the day of the week.	6/15/2009 1:45:30 PM -> Mon (en-US) 6/15/2009 1:45:30 PM -> Пн (ru-RU) 6/15/2009 1:45:30 PM -> lun. (fr-FR)
"dddd"	The full name of the day of the week.	6/15/2009 1:45:30 PM -> Monday (en-US) 6/15/2009 1:45:30 PM -> понедельник (ru-RU) 6/15/2009 1:45:30 PM -> lundi (fr-FR)
"f"	The tenths of a second in a date and time value.	6/15/2009 13:45:30.617 -> 6 6/15/2009 13:45:30.050 -> 0
"ff"	The hundredths of a second in a date and time value.	6/15/2009 13:45:30.617 -> 61 6/15/2009 13:45:30.005 -> 00
"fff"	The milliseconds in a date and time value.	6/15/2009 13:45:30.617 -> 617 6/15/2009 13:45:30.0005 -> 000
"ffff"	The ten thousandths of a second in a date and time value.	6/15/2009 13:45:30.6175 -> 6175 6/15/2009 13:45:30.00005 -> 0000
"fffff"	The hundred thousandths of a second in a date and time value.	6/15/2009 13:45:30.61754 -> 61754 6/15/2009 13:45:30.000005 -> 00000
"ffffff"	The millionths of a second in a date and time value.	6/15/2009 13:45:30.617542 -> 617542 6/15/2009 13:45:30.0000005 -> 000000
"fffffff"	The ten millionths of a second in a date and time value.	6/15/2009 13:45:30.6175425 -> 6175425 6/15/2009 13:45:30.0001150 -> 0001150
"F"	If nonzero, the tenths of a second in a date and time value.	6/15/2009 13:45:30.617 -> 6 6/15/2009 13:45:30.050 -> (no

Format specifier	Description	Examples
		output)
"FF"	If nonzero, the hundredths of a second in a date and time value.	6/15/2009 13:45:30.617 -> 61 6/15/2009 13:45:30.005 -> (no output)
"FFF"	If nonzero, the milliseconds in a date and time value.	6/15/2009 13:45:30.617 -> 617 6/15/2009 13:45:30.0005 -> (no output)
"FFFF"	If nonzero, the ten thousandths of a second in a date and time value.	6/1/2009 13:45:30.5275 -> 5275 6/15/2009 13:45:30.00005 -> (no output)
"FFFFF"	If nonzero, the hundred thousandths of a second in a date and time value.	6/15/2009 13:45:30.61754 -> 61754 6/15/2009 13:45:30.000005 -> (no output)
"FFFFFF"	If nonzero, the millionths of a second in a date and time value.	6/15/2009 13:45:30.617542 -> 617542 6/15/2009 13:45:30.0000005 -> (no output)
"FFFFFFF"	If nonzero, the ten millionths of a second in a date and time value.	6/15/2009 13:45:30.6175425 -> 6175425 6/15/2009 13:45:30.0001150 -> 000115
"g", "gg"	The period or era.	6/15/2009 1:45:30 PM -> A.D.
"h"	The hour, using a 12-hour clock from 0 to 11.	6/15/2009 1:45:30 AM -> 1 6/15/2009 1:45:30 PM -> 1
"hh"	The hour, using a 12-hour clock from 00 to 11.	6/15/2009 1:45:30 AM -> 01 6/15/2009 1:45:30 PM -> 01
"H"	The hour, using a 24-hour clock from 0 to 23.	6/15/2009 1:45:30 AM -> 1 6/15/2009 1:45:30 PM -> 13
"HH"	The hour, using a 24-hour clock from 00 to 23.	6/15/2009 1:45:30 AM -> 01 6/15/2009 1:45:30 PM -> 13
"K"	Time zone information.	6/15/2009 1:45:30 PM, Kind Unspecified -> 6/15/2009 1:45:30 PM, Kind Utc -> Z 6/15/2009 1:45:30 PM, Kind Local -> -07:00
"m"	The minute, from 0 through 59.	6/15/2009 1:09:30 AM -> 9 6/15/2009 1:09:30 PM -> 9
"mm"	The minute, from 00 through 59.	6/15/2009 1:09:30 AM -> 09 6/15/2009 1:09:30 PM -> 09

Format specifier	Description	Examples
"M"	The month, from 1 through 12.	6/15/2009 1:45:30 PM -> 6
"MM"	The month, from 01 through 12.	6/15/2009 1:45:30 PM -> 06
"MMM"	The abbreviated name of the month.	6/15/2009 1:45:30 PM -> Jun (en-US) 6/15/2009 1:45:30 PM -> juin (fr-FR) 6/15/2009 1:45:30 PM -> Jun (zu-ZA)
"MMMM"	The full name of the month.	6/15/2009 1:45:30 PM -> June (en-US) 6/15/2009 1:45:30 PM -> juni (da-DK) 6/15/2009 1:45:30 PM -> uJuni (zu-ZA)
"s"	The second, from 0 through 59.	6/15/2009 1:45:09 PM -> 9
"ss"	The second, from 00 through 59.	6/15/2009 1:45:09 PM -> 09
"t"	The first character of the AM/PM designator.	6/15/2009 1:45:30 PM -> P (en-US) 6/15/2009 1:45:30 PM -> 午 (ja-JP) 6/15/2009 1:45:30 PM -> (fr-FR)
"tt"	The AM/PM designator.	6/15/2009 1:45:30 PM -> PM (en-US) 6/15/2009 1:45:30 PM -> 午後 (ja-JP) 6/15/2009 1:45:30 PM -> (fr-FR)
"y"	The year, from 0 to 99.	1/1/0001 12:00:00 AM -> 1 1/1/0900 12:00:00 AM -> 0 1/1/1900 12:00:00 AM -> 0 6/15/2009 1:45:30 PM -> 9
"yy"	The year, from 00 to 99.	1/1/0001 12:00:00 AM -> 01 1/1/0900 12:00:00 AM -> 00 1/1/1900 12:00:00 AM -> 00 6/15/2009 1:45:30 PM -> 09
"yyy"	The year, with a minimum of three digits.	1/1/0001 12:00:00 AM -> 001 1/1/0900 12:00:00 AM -> 900 1/1/1900 12:00:00 AM -> 1900 6/15/2009 1:45:30 PM -> 2009
"yyyy"	The year as a four-digit number.	1/1/0001 12:00:00 AM -> 0001 1/1/0900 12:00:00 AM -> 0900 1/1/1900 12:00:00 AM -> 1900 6/15/2009 1:45:30 PM -> 2009

Format specifier	Description	Examples
"yyyyy"	The year as a five-digit number.	1/1/0001 12:00:00 AM -> 00001 6/15/2009 1:45:30 PM -> 02009
"z"	Hours offset from UTC, with no leading zeros.	6/15/2009 1:45:30 PM -07:00 -> -7
"zz"	Hours offset from UTC, with a leading zero for a single-digit value.	6/15/2009 1:45:30 PM -07:00 -> -07
"zzz"	Hours and minutes offset from UTC.	6/15/2009 1:45:30 PM -07:00 -> -07:00
":"	The time separator.	6/15/2009 1:45:30 PM -> : (en-US) 6/15/2009 1:45:30 PM -> . (it-IT) 6/15/2009 1:45:30 PM -> : (ja-JP)
"/"	The date separator.	6/15/2009 1:45:30 PM -> / (en-US) 6/15/2009 1:45:30 PM -> - (ar-DZ) 6/15/2009 1:45:30 PM -> . (tr-TR)
"string" 'string'	Literal string delimiter.	6/15/2009 1:45:30 PM ("arr:" h:m t) -> arr: 1:45 P 6/15/2009 1:45:30 PM ('arr:' h:m t) -> arr: 1:45 P
%	Defines the following character as a custom format specifier.	6/15/2009 1:45:30 PM (%h) -> 1
\	The escape character.	6/15/2009 1:45:30 PM (h \h) -> 1 h
Any other character	The character is copied to the result string unchanged.	6/15/2009 1:45:30 AM (arr hh:mm t) -> arr 01:45 A

9 Appendix D: SetTimeZone Field Usage

- **Bias**

  The current bias for local time translation on this computer, in minutes. The bias is the difference, in minutes, between Coordinated Universal Time (UTC) and local time. All translations between UTC and local time are based on the following formula:

  UTC = local time + bias

  This field is required.

- **StandardDate**

  A SYSTEMTIME structure ([MS-DTYP] section 2.3.11) that contains a date and local time when the transition from daylight saving time to standard time occurs on this operating system. If the time zone does not support daylight saving time or if the caller needs to disable daylight saving time, the **wMonth** member in the SYSTEMTIME structure must be zero. If this date is specified, the **DaylightDate** field must also be specified. Otherwise, the system assumes that the time zone data is invalid and no changes will be applied.

  To select the correct day in the month, set the **wYear** member to zero, the **wHour** and **wMinute** members to the transition time, the **wDayOfWeek** member to the appropriate weekday, and the **wDay** member to indicate the occurrence of the day of the week within the month (1 to 5, where 5 indicates the final occurrence during the month if that day of the week does not occur 5 times).

  Using this notation, specify 02:00 on the first Sunday in April as follows: **wHour** = 2, **wMonth** = 4, **wDayOfWeek** = 0, **wDay** = 1. Specify 02:00 on the last Thursday in October as follows: **wHour** = 2, **wMonth** = 10, **wDayOfWeek** = 4, **wDay** = 5.

  If the **wYear** member is not zero, the transition date is absolute; it will occur only one time. Otherwise, it is a relative date that occurs yearly.

- **StandardBias**

  The bias value to be used during local time translations that occur during standard time. This field is ignored if a value for the **StandardDate** field is not supplied.

  This value is added to the value of the **Bias** field to form the bias used during standard time. In most time zones, the value of this field is zero.

- **DaylightDate**

  A SYSTEMTIME structure that contains a date and local time when the transition from standard time to daylight saving time occurs on this operating system. If the time zone does not support daylight saving time or if the caller needs to disable daylight saving time, the **wMonth** member in the SYSTEMTIME structure must be zero. If this date is specified, the **StandardDate** field must also be specified. Otherwise, the system assumes that the time zone data is invalid and no changes will be applied.

  To select the correct day in the month, set the **wYear** member to zero, the **wHour** and **wMinute** members to the transition time, the **wDayOfWeek** member to the appropriate weekday, and the **wDay** member to indicate the occurrence of the day of the week within the month (1 to 5, where 5 indicates the final occurrence during the month if that day of the week does not occur 5 times).

If the **wYear** member is not zero, the transition date is absolute; it will occur only one time. Otherwise, it is a relative date that occurs yearly.

- **DaylightBias**

  The bias value to be used during local time translations that occur during daylight saving time. This field is ignored if a value for the **DaylightDate** field is not supplied.

  This value is added to the value of the **Bias** field to form the bias used during daylight saving time. In most time zones, the value of this field is -60.

10 Appendix E: PROPVARIANT

Propvariant type	Code	Propvariant member	Value representation
VT_EMPTY	0	None	A property with a type indicator of **VT_EMPTY** has no data associated with it; that is, the size of the value is zero.
VT_NULL	1	None	This is like a pointer to NULL.
VT_I1	16	**cVal**	A 1-byte signed integer.
VT_UI1	17	**bVal**	A 1-byte unsigned integer.
VT_I2	2	**iVal**	Two bytes representing a 2-byte signed integer value.
VT_UI2	18	**uiVal**	A 2-byte unsigned integer.
VT_I4	3	**lVal**	A 4-byte signed integer value.
VT_UI4	19	**ulVal**	A 4-byte unsigned integer.
VT_INT	22	**intVal**	A 4-byte signed integer value (equivalent to **VT_I4**).
VT_UINT	23	**uintVal**	A 4-byte unsigned integer (equivalent to **VT_UI4**).
VT_I8	20	**hVal**	An 8-byte signed integer.
VT_UI8	21	**uhVal**	An 8-byte unsigned integer.
VT_R4	4	**fltVal**	A 32-bit IEEE floating point value.
VT_R8	5	**dblVal**	A 64-bit IEEE floating point value.
VT_BOOL	11	**boolVal** (**bool** in earlier designs)	A Boolean value, a WORD that contains 0 (FALSE) or -1 (TRUE).
VT_DATE	7	**date**	A 64-bit floating point number representing the number of days (not seconds) since December 31, 1899. For example, January 1, 1900, is 2.0, January 2, 1900, is 3.0, and so on). This is stored in the same representation as **VT_R8**.
VT_CLSID	72	**puuid**	A pointer to a class identifier (CLSID) (or other globally unique identifier (GUID)).
VT_LPWSTR	31	**pwszVal**	A pointer to a null-terminated Unicode string in the user default locale.
VT_VECTOR	0x1000	**ca***	If the type indicator is combined with **VT_VECTOR** by using an OR operator, the value is one of the counted array values. This creates a DWORD count of elements, followed by a pointer to the specified repetitions of the value. For example, a type indicator of **VT_LPSTR**|**VT_VECTOR** has a DWORD element count, followed by a pointer to an array of LPSTR elements. **VT_VECTOR** can be combined by an OR operator with the following types: **VT_I1**, **VT_UI1**, **VT_I2**, **VT_UI2**, **VT_BOOL**, **VT_I4**, **VT_UI4**, **VT_R4**, **VT_R8**, **VT_ERROR**,

Propvariant type	Code	Propvariant member	Value representation
			VT_I8, **VT_UI8**, **VT_CY**, **VT_DATE**, **VT_FILETIME**, **VT_CLSID**, **VT_CF**, **VT_BSTR**, **VT_LPSTR**, **VT_LPWSTR**, and **VT_VARIANT**. **VT_VECTOR** can also be combined by an OR operation with **VT_BSTR_BLOB**, however it is for system use only.

11 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

12 Index

S

T

V