ANSI E1.17 - 2015 (R2020) Architecture for Control Networks – ACN Architecture

Part of ANSI E1.17 – 2015 (R2020) approved by the ANSI Board of Standards Review on 23 March 2020.
This part has no substantive changes from the 2010 edition.

TSP document ref. CP/2003-1007-R4-draft-439:454M

ANSI E1.17-2015 (R2020) Architecture for Control Networks – ACN Architecture

TSP document ref. CP/2003-1007-R4-draft-439:454M

	Revision History
Revision R4-draft-439:454M	2011-04-29
Revision R4pre2	2005-10-11
Revision R4pre1	2005-08-02
Revision R3	2004-11-09
Revision R2	2004-10-07
Revision R1	2003

Abstract

E1.17 "Architecture for Control Networks" (ACN) consists of a suite of protocols and languages which may be configured and combined with other standard protocols in a number of ways to form flexible networked control systems.

This document includes an informative introduction which describes the origins and aims of the protocol suite and gives examples of how the parts can be put together.

Following the introduction, the concepts and specifications which are needed within ACN to support the protocols of the ACN architecture but which are not a part of those individual protocols are defined. These include the common packet format used across several ACN protocols, definitions of components and use of addressing in ACN, the Root Layer protocol which interfaces to the underlying transport, the arrangement of protocol layering in ACN systems and use of interoperability profiles.

Table of Contents

1. Informative Introduction: Origins, Aims and Design Reasoning

Non-normative

The Architecture for Control Networks arose from a need within the entertainment technology industry in general and particularly the lighting industry for a common interoperable protocol or protocols for control of equipment which would take advantage of the new technologies which were gaining dominance and offering cheap high speed communications in many areas.

This section describes the background from which ACN was developed, the design reasoning and processes which have gone into generating ACN and the way a typical implementation fits together. It is informative only, but is recommended reading as starting point for those new to ACN.

In designing ACN, great reliance has been placed on a wide variety of existing technologies which are constantly changing and progressing. The protocols of ACN have well defined functionality and clear boundaries and layering. It follows that all these pieces can and will be put together in different ways to suit different applications. The things which tie all these pieces together are interoperability profiles which specify which pieces must be used and what their operational parameters must be to achieve interoperability over a particular application domain. They may also specify what form of bridging or translation is required to connect them to networks using different profiles.

Thus the ACN architecture consists of a number of separate specifications for protocol formats, languages etc., together with interoperability profiles which specify where and how those specifications are to be used and call up external specifications as necessary. The driving force for ACN has been control of entertainment technology equipment across mainstream networks such as Ethernet and the initial interoperability profiles and specifications reflect this. However, other applications can be developed within this framework.

1.1. Background

The following summarizes the situation at the beginning of the ACN project:

1.1.1. The Need for a New Control Protocol

The lighting industry has been served well since the late 1980s by the USITT DMX512 data link standard for interconnecting controllers with dimmers [DMX512]. Other entertainment technologies have other data transport standards both proprietary and open e.g. MIDI Show control, MIDI Machine control, PA-422 and a plethora of proprietary systems in the audio industry. However, a lot has changed since these were introduced and in particular the explosive growth of sophisticated networking since the late 1990s.

1.1.1.1. Needs are greater

Typical lighting systems integrate the control of a large and growing variety of automated fixtures, color scrollers, and other devices where previously there were just dimmers. Furthermore, the lighting control system is expected to integrate with systems controlling sound, automation, environmental equipment, pyrotechnics and a host of other things. With the rise of themed environments of various kinds, the line between performance control and architectural and environmental control has become blurred, if not erased.

1.1.1.2. Capabilities are greater

Along with the phenomenal growth of business LANs and the global Internet have come a variety of powerful networking technologies and general-purpose protocols available off-the-shelf. Meanwhile, control consoles and processor units are using embedded computers with mainstream operating systems and hardware, capable of supporting modern networks. The technology is ready for the application of advanced computer networks to entertainment technology control.

1.1.1.3. Expectations are greater

Professionals in entertainment technology use computers, LANs, and the Internet, both at work and at home. They know what computers and computer networks are capable of, and they expect as much from their control systems. Standardization has been a key force in mainstream computing, allowing an enormous variety of equipment from thousands of different manufacturers to "plug-and-play". This is a model of what is possible, and users of lighting equipment wonder, quite rightly, why a similar model cannot be realized in lighting control. The ESTA Architecture for Control Networks (E1.17) [ACN] is an effort to address these needs, capabilities, and expectations.

To ensure continued success and growth, the industry needs flexible, scalable, and powerful industry-standard mechanisms for discovery, configuration, and control. The challenge that ACN meets is to provide that power and flexibility while still enabling lightweight devices to participate.

1.1.2. Design Goals

To guide the development of ACN a set of design goals for an control network were enumerated at the start of the project. These design goals serve as a set of minimum requirements against which ACN can be judged.

1. **Interoperability across manufacturers.** The protocol should enable various participating manufacturers' equipment to communicate usefully. e.g. to let one manufacturer's console control another manufacturer's dimmers or moving lights.

2. **Many sources, many sinks.** The protocol should provide for multiple sources of control data on the same network, as well as multiple consumers of that data. Sources and sinks should not be equated with controllers and controlled devices - data will flow both ways.

3. **One "wire", but multiple, independent uses.** The protocol should allow a single network to support multiple independent uses. For example, in large, complex installations it will be desirable to dynamically configure sub-venues as independent universes of control, with independent addressing, etc. Another example: while integration of audio and lighting control may be possible, it is not always desirable and the two systems would frequently be operated independently.

4. **A mainstream protocol.** The protocol should be capable of traveling over mainstream network protocols. It should not attempt to reinvent the functionality of lower-layer network protocols.

5. **Maximum opportunity to use off-the-shelf technology.** The protocol should be designed to exploit the enormous variety of off-the-shelf networking hardware and software (routers, switches, protocol stacks, diagnostic tools, etc.) available from third parties. Moreover, it should be designed so that it can continue to "ride the wave" of the rapidly evolving state of the art in networking in the future.

6. **Support for manufacturer-specific uses.** Only a subset of the control and data requirements of a modern entertainment technology installation will be standardized industry-wide. The protocol should support manufacturer-specific extensions in an elegant way. The subset that is standardized should not be a conceptual orphan, but should fit in naturally with the protocol as a whole.

7. **Scalability.** The protocol should adequately address the needs of the full range of size and complexity of applications. It should be simple enough for use on the smallest systems, and also scale up to networks controlling large Broadway Musicals, hotel complexes, theme parks, etc.

8. **Extensibility.** The protocol should be designed to be easily and cleanly extended over time to handle future requirements as they arise. The protocol must be "built to last" and "future proof".

9. **Ease of configuration.** The protocol should provide a means for the network to be easily configured and managed. Devices should dynamically discover one another and one another's capabilities without user intervention: "plug-and-play".

10. **Efficient and predictable use of available bandwidth.** The protocol should be reasonably frugal with network bandwidth. This requirement follows from design goals 2, 3, 7, 8, above. For example, rather than continually updating the level of a device it should be possible to send only changes in output levels. Moreover, network bandwidth usage should be predictable, so designers of new systems can correctly specify network requirements.

11. **Flexibility and control with respect to sub-networks and routing.** To achieve scalability and efficient use of available bandwidth, it is essential that the protocol does not foreclose the normal range of options used to structure network traffic. E.g., network designers must be free to use sub-net addressing and routers in ways that reflect the actual network structure. Sub-net addressing provides a means for limiting network traffic by isolating sub-nets in complex networks. For example, the protocol should not use sub-net addressing in ways that force all devices of a given type to reside in the same sub-net.

12. **Fault tolerance.** The kinds of applications in which ACN will be utilized cannot tolerate frequent network failures. Nor can end-users be expected to have a high level of networking expertise. Ideally, network failures will be minimized and the network will recover gracefully and with a minimum of user intervention when failures do occur.

1.2. ACN Design Reasoning

This section gives a very brief overview of the design reasoning behind ACN and introduces the main elements of the suite. Much of this section describes the particular use of ACN protocols over TCP/IP for device control using Device Management Protocol. It is important to bear in mind that while this usage drove much of the development, many other configurations are possible and have been considered.

1.2.1. Getting and Setting Properties

Controlling ACN devices is achieved by representing the functions of that device as a set of variables or "properties". The device is monitored by "getting" the values of these properties and controlled by "setting" them. In ACN this most fundamental function is performed by Device Management Protocol (DMP) [DMP] whose most basic messages are Get_property and Set_property. DMP also defines an addressing scheme for variables within a device. DMP does not itself apply any meaning to those properties - a dimmer level is no different in DMP from a telescope azimuth.

1.2.2. Specific Functionality of Controllable Components

DMP provides a generalized method for Getting and Setting values of variables in components. Without knowing what those variables represent, a controller can achieve nothing. It would be possible to use a scheme where properties for specific functions are always at the same address, but this is very unwieldy and fails on goals 7 and 8. In ACN the mapping between properties and specific device function is provided by a separate device description. Device Description Language (DDL) [DDL] defines the format and language for these descriptions in a way that allows a controller to find the functionality of each property. The DDL model not only allows simple controllers to seek out the properties for functions it knows about, such as intensity or gobo number, but also allows smarter controllers to be designed which can work out how to deal with functions which they have never met before.

1.2.3. Components and Identity

In a complex networking environment, there is not a one-to-one correspondence between network interfaces and the devices or processes which are sending or receiving the data e.g. one single device may have multiple interfaces, while a computer with just a single interface may be running two or more completely independent ACN applications. So some mechanism is needed to identify and address the transmitter or receiver of data.

There is also a need for persistent identity of devices independent of network changes. For example, a motion controller needs to relate a particular winch (among many identical ones) to particular actions which the controller has stored for it. In many systems that same winch may be assigned a different network address the next time the system is powered up.

In ACN each distinct endpoint transmitting and receiving ACN data is called a *Component* and all ACN communications take place between *components*. Each ACN component has a *component identifier* or *CID* which is unique not just within the system but across the whole world and does not change with time.

A device such as a dimmer or audio amplifier will typically be a single ACN component, but a large control console or a general purpose computer may contain multiple components.

1.2.4. Packing Multiple Messages and Multicasting

Get/Set_property messages are typically very short, but transports like Ethernet and TCP/IP are most efficient when transferring large blocks of data and become inefficient when packets become very small. To combat this mismatch, ACN allows large numbers of short messages to be packed into a single *packet*. A packet containing multiple messages is then sent to all the devices concerned in a single transmission and those devices extract their own messages. Sending a single packet to a selected group of devices is called "multicasting".

1.2.5. Reliability and Ordering

If a Set_property message to a device goes missing or arrives out of order, that device may be in an incorrect state. A reliable transport which ensures that messages do arrive and in the correct order, or if there is a failure, the controller knows about it can relieve controllers of a large amount of effort in checking that messages have arrived and that the state of the device is as expected. However, in a multicast protocol a straightforward acknowledgement of every message cannot work since the outgoing packet may be addressed to hundreds of devices and if they all acknowledge, the network becomes flooded – so a more sophisticated and efficient reliable multicast technique is required. In ACN this is achieved by Session Data Transport (SDT) [SDT] which can provide reliable, *ordered* delivery of multicast messages.

1.2.6. Discovery

Ease of configuration (goal 9), means that human operators should not have to tell the controller which devices are present and how to control them. The process of automatically finding and organizing devices and working out how to control them is called discovery. In ACN it takes several stages. 1. the controller has to find which ACN devices are on the network. 2. it has to find out what sort of devices they are. 3. it has to find the structure of the devices so that it can control them if required.

In IP networks Service Location Protocol (SLP) [SLPv2] is used for the first stage and the initial part of 2 (discovery of the root device type). Further discovery of device type and stage 3 is done by examination of the DDL description of particular devices.

1.2.7. Layering and Modularity

In accordance with design goals 3, 6 & 8 there is a benefit in separating the reliable multicast technique from the Get/Set_property protocol. This is why SDT, DMP and DDL are separate. It allows alternative protocols for other purposes (e.g. proprietary protocols, show control, time code, file transfer, content streaming...) to use the reliability and group management provided by SDT. It allows DMP to be used over other transports if the SDT model is not appropriate. Finally and most important, dividing the protocol into layers simplifies it because each layer is only concerned with a well defined part of the problem.

1.2.8. Efficiency Tradeoffs, Optimizations

1.2.8.1. Common Packet Format

In developing SDT and DMP it was recognized that the messages in both protocols have many similarities and that by using a common message format across all the ACN protocols, the code to recognize and decode messages can be shared across different message types. This is why ACN defines a common message format the *Protocol Data Unit* or *PDU*.

1.2.8.2. Message Packing

A network protocol can be optimized for (among other things) packet size, processing speed or processing code size. Any such optimization involves trade-offs. Goal 10 of ACN is efficient use of bandwidth. The ACN PDU format provides a number of optimizations to allow the number of messages packed into a single *packet* to be maximized. This involves a small increase in processing complexity but analysis has shown packing improvements of a factor of four or more in some common circumstances. Furthermore the small increase in processing complexity is offset by the efficiency gain and simplicity provided by using a common PDU format across all PDUs and layers.

1.2.8.3. Range Addressing

At the DMP layer, the addressing scheme and use of range addressing allow very large numbers of property values to be transferred with minimal overhead for transmitting their addresses. This adds to the bandwidth efficiency of DMP based systems.

1.2.9. Root Layer Protocol

Because of the way the common packet format is used, the modularity of the protocols and the desire to allow a common entry point in the stack for all ACN protocols, the Root Layer protocol is defined. The Root Layer Protocol specifies how PDUs from different ACN protocols may be combined into packets for transmission via the transport, and how they are separated and passed for processing on receipt.

1.2.10. Transporting ACN protocols - use of TCP/IP

Careful separation of the ACN modules allows it to operate over many transports. However, initial LAN applications of ACN are expected to use TCP/IP and this provides a good model for future use with other transports.

The near universal acceptance of the TCP/IP family of protocols means that there exist an enormous number of commercially available hardware and software products for the implementation of such networks. Moreover, the continuing commercial momentum behind TCP/IP means that as a platform it will continue to evolve and be supported for the foreseeable future. These kinds of considerations make a compelling case for using the TCP/IP protocols, or a suitable subset of them, as the lower-level protocols within ACN systems.

1.2.10.1. Flexibility in the choice of network media

TCP/IP provides an abstraction of a network so that application programs that communicate via TCP/IP generally do not care, or even have to know, what the underlying network technology is. Technologies commonly used at present for carrying TCP/IP include IEEE 802.3 (wired Ethernet) at a variety of data rates, IEEE 802.11 (wireless Ethernet or “WiFi”), IEEE 1394 (Firewire), modems and other serial links and high speed links for “backbone” distribution such as ATM, SONET and FDDI. As new technologies emerge and TCP/IP is adapted to them, so applications that communicate via TCP/IP will be portable to these new network technologies. Thus when run on TCP/IP, ACN protocols are not Ethernet protocols, but enjoy the full benefits of infrastructure and tools created for TCP/IP networks. Because of its modularity, ACN will not be made obsolete by the appearance of new network technologies. Instead, applications that employ ACN will be positioned to “ride the wave” of advances in network technology.

1.2.10.2. What we do not get from TCP/IP

There are currently no reliable multicast protocols that suit the needs of the ACN architecture, therefore we have developed SDT for reliable multicast and session management in ACN.

1.2.10.3. ACN Subset of TCP/IP

ACN only uses a subset of the large TCP/IP suite and every attempt is made to keep this subset as small and simple as possible. In particular operation of SDT over UDP [UDP] is possible without any use of the complex TCP protocol itself (Confusingly, TCP is just one optional part of the TCP/IP suite although it has given its name to the whole suite).

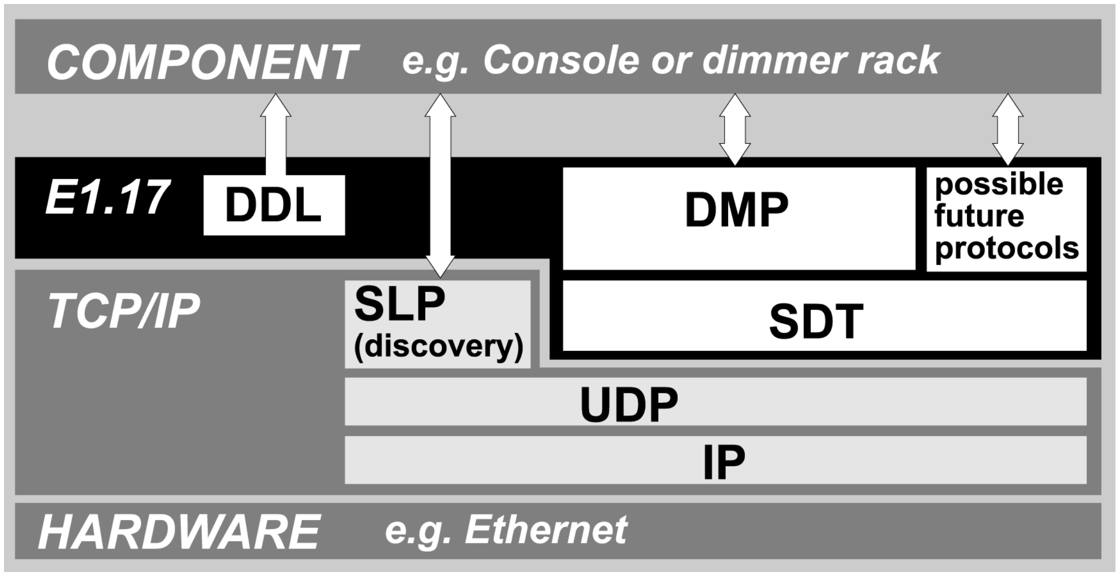
1.2.10.4. ACN With Other Transports

In the interests of good modular protocol design, the interface between ACN protocols and TCP/IP is clean and well defined and it is quite possible to separate ACN from TCP/IP where it is required to operate over alternative transports. Maintaining this division will also assist in addressing any issues with migration from version 4 to version 6 and above of the TCP/IP suite.

1.2.11. Operation of a Typical System

In an ACN system for control of devices over Ethernet, a controller will send Get/Set_property messages to devices as defined by DMP. These messages are transmitted using SDT which provides reliability, online status and management of groups of devices. All DMP and SDT messages are packed in individual PDUs using the common PDU format. These are then transmitted using UDP over Ethernet. New devices coming online are initially discovered using SLP and their specific functionality is then analyzed and configured using DDL.

Figure 1. ACN Modules


COMPONENT e.g. Console or dimmer rack
E1.17
DDL
DMP
possible future protocols
TCP/IP
SLP (discovery)
SDT
UDP
IP
HARDWARE e.g. Ethernet

Note: It is important to bear in mind that while this usage drove much of the development, many other configurations are possible and have been considered.

1.2.12. ACN in Other Environments and Uses - Interoperability Profiles

The protocols resulting from the ACN project have followed a very modular design process and every attempt has been made to make modules independent of one another - this allows modules to be added, omitted or recombined as required. Furthermore within modules many parameters are not fixed but are necessarily dependent for optimal operation on the settings of other modules or on environmental factors such as network topology, latency or reliability requirements.

Because of this the core protocol suite does not define fully how interoperability is to be achieved in any particular application area. This problem is addressed by producing interoperability profiles which define what combinations of modules and parameters must be used to gain interoperable operation in a particular environment (e.g. for a Local Area Network operating over UDP/IP on Ethernet). Interoperability profiles will also spell out what is needed to link pieces of equipment which conform to other profiles - this might be as simple as bridging one physical network type to another (e.g. Ethernet to IEEE 1394 - “Firewire”) or may require more complex processing (e.g. when building a proxy router between an Ethernet/IP network and an RS485 serial implementation).

Interoperability profiles are separate from the protocol suite itself but are required to define an implementation. The section “ACN Operation In a Nutshell” above actually describes an interoperability profile for control of devices using DMP over SDT over UDP over IPv4 over Ethernet - probably specifically CAT-5 Twisted pair topology.

1.3. Other protocols within the ACN Architecture

It is not expected that ACN will remain a static standard. The current protocols may be extended or refined and new protocols may be added within the overall architecture.

1.3.1. Recursive use of PDU Format in Other Protocols

The payload of root layer PDUs are passed on to higher layer protocols and it is entirely up to those protocols to specify the format and use of the data. However, within ACN as currently specified, there is just one higher layer protocol defined which interfaces directly to the root layer: Session Data Transport [SDT] is defined within the ACN suite for providing reliable ordered delivery of messages in a multicast context. It not only receives its data from root layer PDUs, but uses the same PDU format internally - with multiple nesting levels - for organizing its messages. In SDT the vector field within the PDU is used to specify the message type or command code.

Device Management Protocol [DMP] is the protocol specified within ACN to dynamically control things such as lights, mechanical equipment, audio devices etc. DMP is transported by SDT and like SDT, DMP also uses the PDU format for data packing.

This recursive use of a common data packing format allows efficient implementation of ACN systems in small devices because, a single body of code can be optimized and debugged and then reapplied at all layers.

2. Normative Specification

2.1. ACN Components and Component Identifiers (CIDs)

There is not a one-to-one relationship between data sources/sinks and network interfaces. Firstly a single piece of equipment quite commonly has multiple network interfaces either because it supports more than one network medium (e.g. Ethernet and a modem connection) or because it is connected to more than one network segment; and secondly a single computer may be running two or more independent programs using ACN, which have no specific knowledge of each other and yet are required to share one network interface.

An ACN *Component* is a distinct endpoint transmitting and receiving ACN data and all ACN communications take place between components.

Every ACN Component shall be identified by a *Component identifier* or *CID*. A CID shall be a UUID [UUID] which is a 128-bit number which is unique across space and time. Each piece of equipment should maintain the same CID for its entire lifetime (e.g. by storing it in read-only memory). This means that a particular component on the network can be identified as the same thing from day to day despite network interruptions, power down and so on. However, in some systems there may be situations in which volatile components are dynamically created “on the fly” and in these cases the controlling process can generate CIDs as required. The choice of UUIDs for CIDs allows them to be generated as required without reference to any registration process or authority.

2.1.1. Byte Ordering of CIDs

When CIDs are transmitted within ACN protocols (e.g. SDT), they shall be ordered in network byte order as specified in section Section 2.4.6, “Byte Ordering of PDU Fields” and section 4.1.2 of [UUID].

2.2. Protocol Data Units and Blocks - the Standard ACN Message Format

2.2.1. PDUs

Protocols within the ACN suite use a common format which generalizes the structure of a *message*, command or data item as a *Protocol Data Unit* (*PDU*). The information within a PDU and its format are defined below.

2.2.2. PDU Blocks

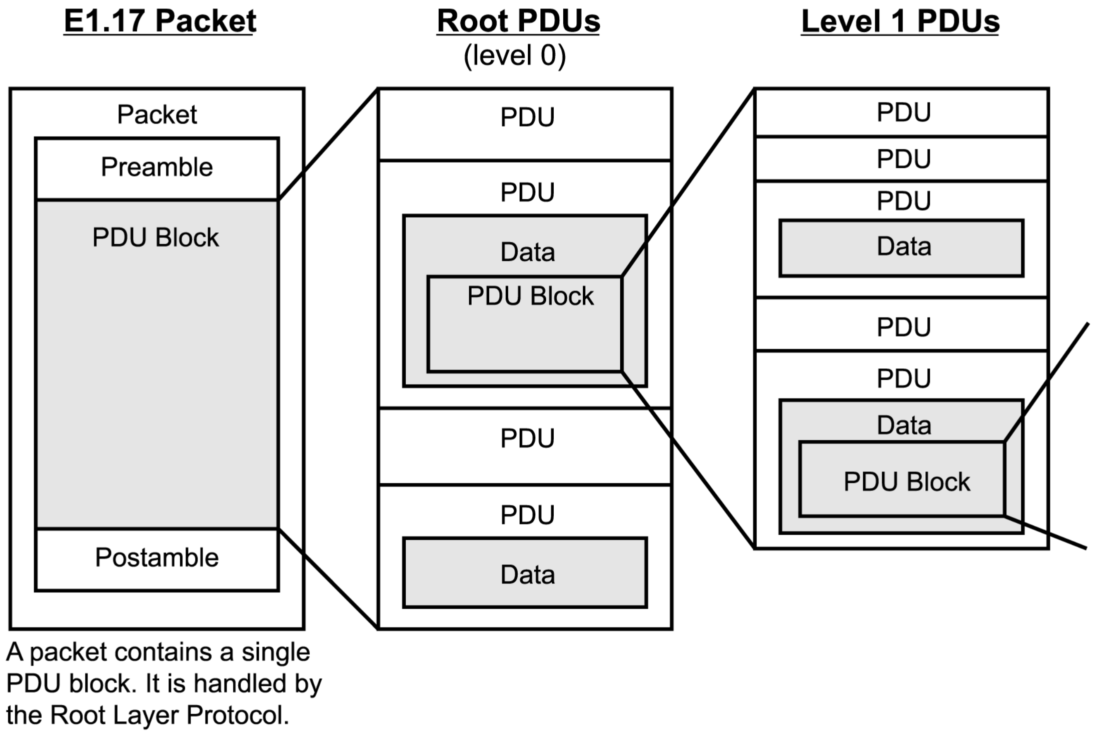
Multiple PDUs may be packed together contiguously in a single data block which is called a *PDU block*.

Within ACN the term *PDU block* shall mean a contiguous set of zero or more PDUs with no intervening non-PDU data, which are packed together according to the packing rules defined below. In order to process a PDU block, its size or the number of PDUs within it must be provided by the context in which it occurs.

2.2.3. Nesting PDU blocks within PDUs

Each PDU in a PDU block includes a data section (see below) and this may itself contain one or more PDU blocks - depending on the specification of individual protocols.

Figure 2. Example of PDU Nesting in an ACN Packet



Note

A more concrete example of PDU nesting and usage is given in Section 2.7, "ACN PDU Structure Example"

2.2.4. Use of PDUs and PDU blocks

PDUs and PDU blocks are the basic building block used to make ACN *packets*.

ACN protocols define messages (or commands) and the rules for their combination or contents and these rules dictate how the PDU structure is used and combined to do the work - the structure is generic but the rules and usage are specific to individual protocols.

2.2.5. Mixing and layering of PDUs and Protocols

PDUs of multiple protocols may appear within a single PDU block subject to any restrictions specified with individual protocols. Protocols may also use PDU nesting for packing data within the single protocol (both SDT and DMP do this).

An ACN protocol may specify within which other PDUs its defined messages (themselves PDUs) may be nested. A protocol may also specify exactly what messages may be embedded within its message PDU blocks. This allows the protocol to specify more exactly how its part of ACN packets are structured.

Protocols carried within the ACN architecture may or may not use the ACN Packet Format for their data (e.g. to encapsulate individual commands). Because the Root Layer Protocol uses PDUs to pack data into a packet it follows that all protocol data within an ACN packet must be enclosed (nested) directly or indirectly within a PDU at the root layer and may additionally be nested at higher layers.

2.2.6. Packets

Within ACN, the term *packet* shall mean a datagram which is carried by an underlying transport protocol and is not wrapped within any outer layer ACN PDU. Packets are transmitted and received by the Root Layer Protocol as defined below.

2.2.7. PDU Block reception and order of PDU processing

On receipt of a PDU block, all PDUs within that block shall be examined and processed strictly in the order in which they occur in the block. In this context, processing refers to unpacking and computing field values and following all E1.17 rules for handling PDU Blocks and PDUs as specified below (see Section 2.4, “PDU Fields” and Section 2.5, “PDU Packing Rules”). ACN protocols shall also carry out further protocol specific processing of PDUs strictly in order in which they occur in the block unless otherwise specified.

2.3. Use of the Transport Protocol

Communication in an ACN system takes the form of packets traveling from one component to one or more other components. These ACN packets must be sent using some underlying protocol - the transport protocol (e.g. UDP). Passing the ACN packet to the transport protocol does this. In addition to the Packet, the Transport protocol will require information such as the following:

- The transport protocol address of the receiver of the ACN packet;
- The transport protocol address of the sender of the ACN packet;
- The length of the ACN packet;
- The data that makes up the packet.

The addresses used by the transport protocol are dependent on that protocol. See annexes and EPIs for details of using ACN on specific protocols.

2.3.1. Transport Protocol Addressing and ACN Filtering

Because underlying transport protocols have no concept of ACN components, an association is required between an ACN component and a transport protocol address. This association does not guarantee that the transport protocol delivers the packet exclusively to a single component and indeed some parts of ACN rely on multicasting for delivery to multiple destinations at once. Thus an ACN implementation has to filter PDUs received and each component can expect to reject many PDUs which are not intended for it. However, ACN protocols should use the addressing mechanisms of the transport protocol to minimize the set of recipients wherever possible. This process is described more fully in [SDT].

2.3.2. The Root Layer Protocol

The lowest ACN layer above the underlying transport (e.g. the contents of a UDP packet) is handled by the *ACN Root Layer Protocol*. The Root Layer protocol may be tailored to fit higher ACN protocols onto each individual transport.

2.4. PDU Fields

Each PDU in a PDU block shall consist of 5 fields: Flags, Length, Vector, Header and Data. With the exception of Flags and Length which are packed together, all fields shall occupy an integer number of octets (which may be zero).

A PDU is laid out in octets as follows:

Figure 3. Layout of PDU

```
 7   6   5   4   3   2   1   0
+---------------------------------+
| flags         |   LengthH       |
| L   V   H   D |   (4 bits)      |
+---------------------------------+
|          LengthL                |
|                                 |
+---------------------------------+
|          LengthX                |
| (only present if L flag = 1)    |
+---------------------------------+
|          Vector                 |
| Length up to 4 octets           |
| as defined by protocol          |
               ...
+---------------------------------+
|          Header                 |
| Length variable                 |
| as defined by protocol          |
             ...
|                                 |
+---------------------------------+
|          Data                   |
| Length variable                 |
| as defined by protocol          |
             ...
|                                 |
+---------------------------------+
```

2.4.1. Flags

Flags is a 4-bit field containing flags L, V, H and D which declare how the PDU is packed as defined below.

2.4.2. Length

Length shall specify the length in octets of the entire PDU including Flags, Length, Vector, Header and Data fields as it occurs in the packet after all packing has been applied. i.e. Length is the number of octets from the start of this PDU to the start of the next in the block (or to the end of the block). This allows rapid traversal of PDUs in a block.

If the PDU length is 4095 octets or less, the L flag shall be set to zero and the PDU length shall be a 12-bit quantity with bits 11..8 (most significant bits) contained in PDU octet 0 bits 3..0, and length bits 7..0 contained in PDU octet 1. (fields LengthH and LengthL respectively in the diagram above).

If the PDU length is 4096 octets or greater (up to 1048575 octets) the L flag shall be set and the PDU length shall be a 20-bit value, with bits 19..16 (most significant bits) contained in PDU octet 0 bits 3..0, length bits 15..8 contained in PDU octet 1 and length bits 7..0 contained in PDU octet 2. (fields LengthH, LengthL and LengthX respectively in the diagram above).

Note that all allowable PDU lengths have a required encoding – there is never a choice of encoding. A consequence of this is that in an implementation where other restrictions (such as MTU) dictate that length must always be less than 4096 octets, the longer 20-bit length can never occur and length will always occupy 12 bits with L cleared.

2.4.3. Vector

Vector is a short, fixed length field indicating to the receiver, how and where to pass this PDU on for the next level of processing – conceptually it points to a handler for the payload of the PDU. Depending on the rules of individual protocols it may for example identify the message type of the PDU (command code), the protocol decoder for the PDU (protocol identifier) or the recipient data stream (demultiplex channel).

For example in the case of the Root Layer Protocol, Vector identifies an ACN protocol such as SDT and the entire PDU is passed to the handler for that protocol. At SDT's base layer the Vector field holds the message identifier (command code) while within an SDT wrapper, vector indicates the recipient component (by memberID).

The size of Vector shall be no greater than four octets. The size of Vector shall be the same for all PDUs in any PDU block. The specific size of Vector within a PDU block shall be a fixed by the protocol rules defining that PDU block.

2.4.4. Header

Header supplies supplementary information on the content of the PDU. The size, use and format of the Header field depends on individual protocol definitions. The Header field may be of zero size.

To ensure that PDUs may be unpacked efficiently, the protocol rules defining and PDU block shall either specify a constant header size for all PDUs in the block or, if the header is of variable size, shall specify an encoding which allows its size to be readily calculated by an unpacking routine without reference to the content of any other PDU field (for example, by placing the header length in the first octet of the header field). This is necessary so that the division between the header and data fields can always be found by an unpacking routine even if the vector or other parts of the header field are not recognized.

2.4.4.1. Preservation of Protocol Layering

Where layered protocols each use the PDU format, PDU nesting levels need not correspond directly to protocol layers. However, where nesting levels cross protocol layers, (i.e. in any PDU whose data is to be passed to another protocol for processing), “opaque” data belonging to the identified protocol shall be carried entirely in the Data field and Header shall not be used to carry client protocol data. This rule ensures clean separation of protocol layers.

2.4.5. Data

Data holds the content of the PDU and its interpretation is defined by the protocol, message type or other handler identified by the Vector field. The Data field may be of zero size. Because the length of the entire PDU is known and the size of all other fields is known, the length of the data field can always be calculated.

The use of separate Header and Data fields gives more scope for protocols to optimize away information which is repetitive or redundant from one PDU to the next. (see Section 2.5.1, "Suppression and Inheritance of Repeating Field Values").

2.4.6. Byte Ordering of PDU Fields

Length and Vector fields within the ACN PDU format shall always occur in network byte order (big endian).

Byte ordering of Header and Data fields is dependent on individual protocol specifications.

Byte order for all protocols defined as part of the ACN suite shall be network byte order (big endian) unless explicitly stated otherwise. Protocols defined as part of the ACN suite should not use any other byte order for any field unless exceptional reasons prevail.

2.5. PDU Packing Rules

Analysis of the type of data which ACN is dealing with shows repetitive structures at many levels in which certain parts of the information do not change from one item to another. The ACN PDU packing aims to allow efficient compression of the data by identifying parts which repeat from one PDU to the next and allowing all but the first occurrence to be omitted from the packet. This is applied to the Vector, Header and Data fields.

2.5.1. Suppression and Inheritance of Repeating Field Values

The three flags V, H and D represent the state of each of the three fields Vector, Header and Data in the PDU respectively.

When constructing a PDU block, any of the three fields, Vector, Header and Data may be omitted if its value is identical to the same field in the preceding PDU in the block. If the value of a field differs from the value of the same field in the preceding PDU in the block then that field shall be present in the PDU.

If a field is omitted (because its value was identical to that in the preceding PDU), then the corresponding flag shall be set to zero. If the field is present, then the corresponding flag shall be set to one.

On unpacking a PDU block, if the PDU flag corresponding to a field is "1", then the value to be used for the field shall be present in the PDU. If the flag is "0", then no value shall be present in the PDU and an "inherited" value for the same field from the previous PDU shall be used. In either case the value of the field must be kept available in case it is inherited by the next PDU in the same block.

Note that for Header and Data there is a difference between an inherited value (H == 0 or D == 0) and a non-existent value which is indicated by H == 1 or V == 1 and a length of zero. If two or more successive PDUs contain a field of zero length, the second and subsequent PDUs may specify the value explicitly (H or V == 1) or by inheritance with no change to the resulting PDU length.

2.5.2. Inheritance at the Start of a PDU Block

The previous value of a PDU field shall not be inherited over from one PDU block to the next. It follows that at the start of a PDU block before unpacking the first PDU, no previous value is available for any field.

Any PDU containing one or more fields whose value is not available for any reason shall be discarded and the next PDU in the block (if any) processed in turn.

Any field value which is inherited from an unavailable value shall itself defined to be unavailable.

2.5.3. Inheritance from Discarded PDUs

It is legal for a PDU to inherit field values from a preceding PDU which itself has been discarded. This could have occurred, for example, because the vector field is unrecognized or because one of its field values was unavailable. This implies that a receiver implementation must retain and be able to identify any available field values in a preceding PDU even when that PDU was discarded.

2.6. The ACN Root Layer Protocol

The purpose of the Root Layer Protocol in ACN is to join base-level protocols onto the transport layer. To do this it must fulfill two functions:

Multiplexing/Demultiplexing
: Multiple protocols may exist at the lowest(transport) layer of the ACN architecture. For example, in the future both SDT and XYZP (a purely hypothetical protocol) may be used concurrently. In this situation the Root Layer protocol packs PDUs from both of these protocols into packets for transmission (multiplexing) and unpacks them and passes them to the appropriate receiving code on reception (demultiplexing). This allows multiple protocols to coexist on a common underlying transport.

Additional Information
: Different underlying transports may vary wildly in the services they provide. Additional information may be necessary or desirable for successful processing of ACN packets. The Root Layer protocol provides the preamble and postamble fields as places to add this information. Typical information in these fields include packet length, error checking (checksum, CRC etc.), packet identification (e.g., on IPv4 Networks the preamble contains a text string to help identify ACN packets for network diagnostics, see [RLP-UDP]).

The term “Root PDU” or “Root Layer PDU” shall mean a PDU which is extracted from the packet by the Root Layer Protocol and so is not enclosed in any outer PDU.

The Root Layer Protocol may be tailored to fit the ACN Architecture onto a wide variety of different transports which may have different requirements. Therefore, while some basic rules and a generic description are provided here, further rules may be specified for specific transport protocols within the appropriate *Interoperability Profiles*.

2.6.1. Root Layer Packet Format

The format of ACN packet shall consist of a preamble field, a single PDU block and a postamble field.

```
+-------------+
 | preamble    |
 +-------------+
 | PDU block   |
 +-------------+
 | postamble   |
 +-------------+
```

Use of the preamble and postamble fields is specific to individual transport protocol.

2.6.1.1. Preamble

The format and content of the Preamble are dependent on the specific transport protocol and shall be defined in the appropriate interoperability profile. The Preamble may be used to hold any data which are necessary for the particular transport and may be empty.

e.g. on UDP systems, a marker string is placed in the Preamble to enable network analyzers and other diagnostic tools to easily identify ACN packets.

If the specification for transporting ACN on a particular protocol does not explicitly specify data to go in the Preamble field then this field shall be empty.

2.6.1.2. PDU Block

The PDU Block within a packet may contain multiple PDUs of the same protocol and/or PDUs of multiple protocols in any combination.

All the PDUs contained in the PDU Block shall follow the PDU format defined in this standard. How the Data field is formatted in these PDUs is up to the specification of the individual protocols. The protocols also define if the PDU contains a PDU block or other means of sub-dividing the contents of the PDU's Data field.

2.6.1.2.1. Vector Field in Root Layer PDUs

The Vector field of all Root Layer PDUs shall contain the Protocol Identifier of the protocol which generated the Data field within the PDU, formatted as defined in this specification.

2.6.1.2.2. Header Field in Root Layer PDUs

The Header field in Root Layer PDUs shall contain the CID of the component that generated the PDU (the Source CID).

2.6.1.2.3. Data Field in Root Layer PDUs

The Data Field in Root Layer PDUs shall contain protocol data which is opaque to the Root Layer Protocol.

2.6.1.3. Postamble

The format and content of the Postamble are dependent on the specific transport protocol and shall be defined in the appropriate interoperability profile. The Postamble may be used to hold any data which are necessary for the particular transport and may be empty.

e.g. on transport protocols which provide no error checking, the Postamble may be specified to hold a checksum for the packet.

If the specification for transporting ACN on a particular transport protocol does not explicitly specify data to go in the Postamble field then this field shall be empty.

The size of the preceding PDU block and therefore the location of the postamble is not specified here. Therefore any EPI which specifies the use of the postamble field shall also specify how it is located.

2.6.2. Root Layer Protocol Operation

On transmission the Root Layer Protocol shall collate PDUs from higher layer protocols into the PDU Block of a packet. It shall add Preamble and Postamble fields as defined by the specification for the particular transport protocol and shall pass the complete packet to that transport protocol for transmission.

On reception, the Root Layer Protocol shall first perform any validation or filtering operations using the Preamble or Postamble fields which are defined for the specific transport protocol used. If the packet is then accepted, it shall extract the PDUs from the packet and pass them on to appropriate protocol handlers in accordance with the *PDU processing rules*.

The Root Layer Protocol shall make the Source CID from the Header field of each PDU available to the higher layer protocols in whatever way is suitable or necessary for the particular implementation.

If a PDU is received whose ProtocolID (in the Vector field) is not recognized, the Root Layer Protocol may log this case but shall not stop processing the packet. It shall discard the unrecognized PDU and continue with the next (if any). The presence of the Length field in all PDUs allows the next PDU to be found with no further knowledge of the meaning of the current one.

Each Root Layer PDU shall only contain data from a single Component.

2.6.3. Root Layer Addressing

The Root Layer Protocol does not directly handle addressing of packets or PDUs and does not translate between Component Identifiers and transport layer addresses as might be expected - this is left to client protocols.

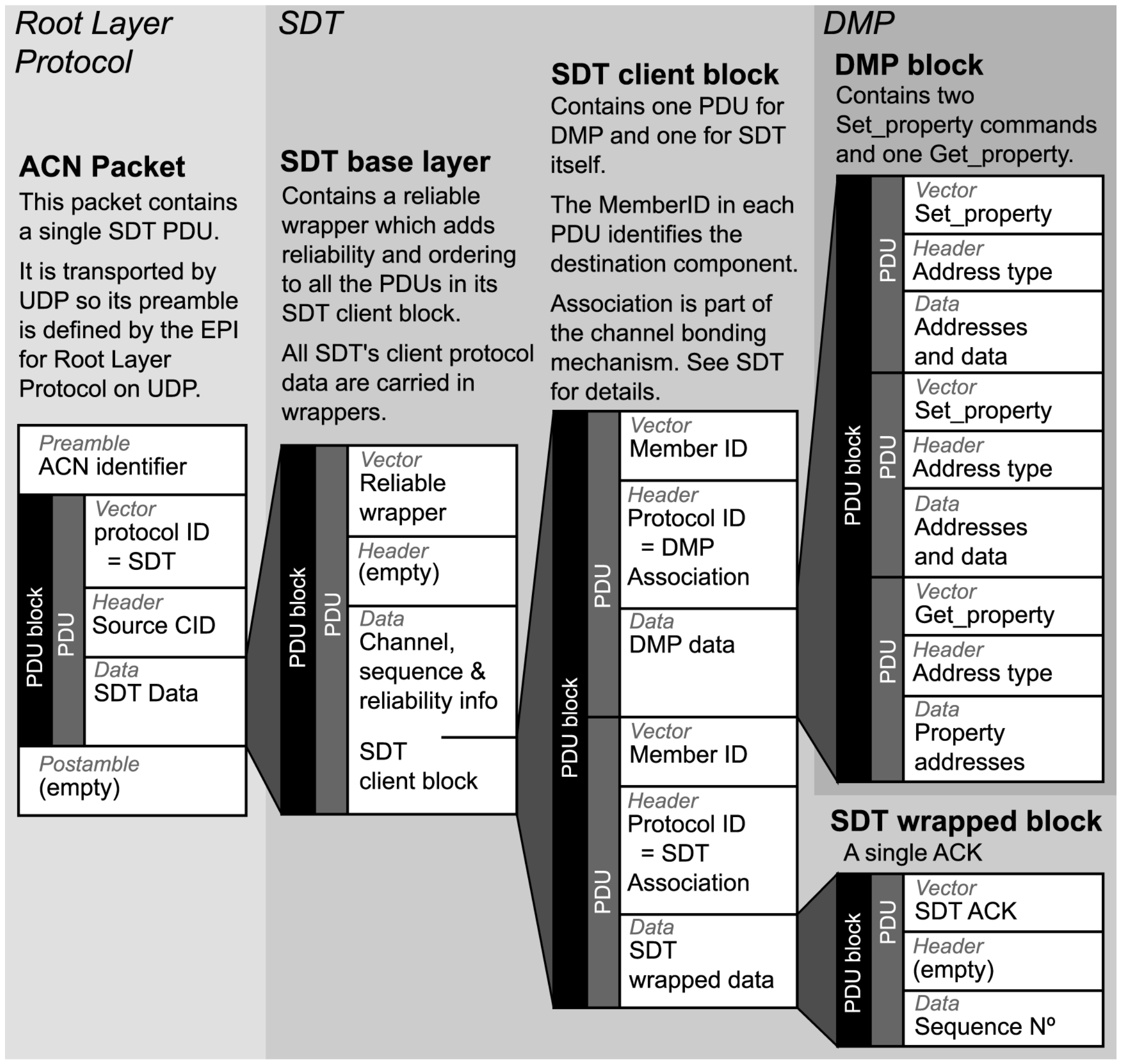
2.6.4. Byte Ordering in Root Layer Protocol

All fields within the ACN Root Layer Protocol PDUs shall use network byte order with the exception of the Data field which is passed to client protocols. The Data field is "opaque" to the Root Layer Protocol and its format - including byte ordering - is defined by the client protocol.

2.7. ACN PDU Structure Example

The example shows the structure of a packet which contains SDT and DMP messages.

Figure 4. Example of PDU Nesting in an ACN Packet


Root Layer Protocol
SDT
DMP
ACN Packet
This packet contains a single SDT PDU.
It is transported by UDP so its preamble is defined by the EPI for Root Layer Protocol on UDP.
Preamble ACN identifier
Vector protocol ID = SDT
Header Source CID
Data SDT Data
Postamble (empty)
PDU block
PDU
SDT base layer
Contains a reliable wrapper which adds reliability and ordering to all the PDUs in its SDT client block.
All SDT's client protocol data are carried in wrappers.
Vector Reliable wrapper
Header (empty)
Data Channel, sequence & reliability info
SDT client block
SDT client block
Contains one PDU for DMP and one for SDT itself.
The MemberID in each PDU identifies the destination component.
Association is part of the channel bonding mechanism. See SDT for details.
Vector Member ID
Header Protocol ID = DMP Association
Data DMP data
Vector Member ID
Header Protocol ID = SDT Association
Data SDT wrapped data
DMP block
Contains two Set_property commands and one Get_property.
Vector Set_property
Header Address type
Data Addresses and data
Vector Set_property
Header Address type
Data Addresses and data
Vector Get_property
Header Address type
Data Property addresses
SDT wrapped block
A single ACK
Vector SDT ACK
Header (empty)
Data Sequence Nº

Note

This diagram is schematic only. It shows the values of the vector, Header and Data fields in each PDU, but does not show the Flags or Length and does not indicate whether values are direct or inherited.

At the two outer packing levels, each PDU block contains just one PDU. The SDT client block contains one PDU addressed to a DMP component and one containing an SDT message (in this case an ACK). Within the PDU passed to the DMP component is a PDU block interpreted by DMP which contains three DMP PDUs.

In accordance with Section 2.4.4, "Header" the size of header is fixed at each level except the DMP PDU block. In the Root Layer Protocol, Header always contains the source CID and is therefore 16 octets. At SDT's base layer and in SDT wrapped PDUs the header is empty (size = 0) while in an SDT client block header always contains a protocolID and association field (6 octets). In DMP the header size varies but is encoded in a consistent way within the first octet.

2.8. Protocol Identifiers

Protocols used within ACN shall be identified within packets using a 32-bit ProtocolID as defined in [ESTA-IDs].

ProtocolIDs shall be transferred in the Root Layer Protocol using all four octets in network byte order.

Unless an different format is explicitly specified within the definition of a higher level protocol, ProtocolIDs shall also be transferred using all four octets in network byte order wherever they occur within ACN messages.

2.9. Interoperability Profiles

The protocols defined in ACN follow a very modular design process and every attempt has been made to make modules independent of one another - this allows modules to be added, omitted or recombined as required. Furthermore within modules many parameters such as timeouts or maximum packet sizes are not fixed by any natural constraint but are necessarily dependent for optimal operation on the settings of other modules or on environmental factors such as network topology, latency or reliability requirements.

Because of this the core protocol suite does not define how interoperability is to be achieved in any particular application area. Barriers to interoperability exist at all levels from the specification of the plug (e.g. 10BASE-T vs 10BASE-2 Ethernet) right up to the highest level protocol used. ACN cannot and should not specify all these parameters as part of its core specification.

This problem is addressed in ACN by specification of Interoperability Profiles which specify what combinations of modules and parameters must be used to gain interoperable operation in a particular environment (e.g. for Device control over a Local Area Network using UDP/IP on Ethernet). An Interoperability profile shall specify values or acceptable ranges for all parameters defined within the ACN protocols it references and shall specify which additional protocols or facilities it relies upon for correct operation.

For example, the profile for Device control over IPv4 over Ethernet must specify which standards must be implemented and combined right from the Ethernet Layer, ancillary IP protocols (IGMP, ICMP, DHCP, ARP, IPv4, RIP, SLP, zeroconf IPv4LL or whatever) and parameters (MTU etc.) through SDT parameters (timeouts, retry counts) up to use of DMP.

An Interoperability Profile should also specify what additional facilities are required to link systems using it to systems operating other related profiles. This might range from a simple link layer bridge (e.g. Ethernet to IEEE 1394 - “Firewire”), to a higher layer protocol translation.

Additional Interoperability Profiles may be defined from scratch or by derivation from existing profiles as required.

2.9.1. Naming of Interoperability Profiles

Each Interoperability Profile shall be given a clearly distinguishable name which may be used in labeling of equipment which operates it. The specification of profiles should be such that users can be confident that two pieces of equipment which correctly implement it can be connected together and will interoperate - subject to any configuration requirements and constraints which must be clearly laid out within that profile.

Definitions

CID	Component Identifier. A UUID [UUID] identifying a particular component.
component	The process, program or application corresponding to a single ACN endpoint. All messages in ACN are sent and received by a component which is identified by a CID. See [Arch] for a more complete definition. See Also CID.
message	A valid PDU.
ordering	A mechanism that ensures that all messages within a session are processed by the members in the order intended by the leader.
packet	A PDU block sent as a distinct unit in the underlying protocol (e.g. a UDP datagram).
PDU	Protocol Data Unit is a single message (or command). A PDU may contain a PDU block with further embedded PDUs.
PDU block	A contiguous set of PDUs (messages) within a packet or containing PDU.

References

Normative

[ACN] Entertainment Services and Technology Association [https://tsp.esta.org]. ANSI E1.17 - 2010,*Entertainment Technology - Architecture for Control Networks*.

[Arch] Entertainment Services and Technology Association [https://tsp.esta.org].. ANSI E1.17 - 2010, *Entertainment Technology – Architecture for Control Networks*. “ACN” Architecture.

[DMP] Entertainment Services and Technology Association [https://tsp.esta.org].. ANSI E1.17 - 2010, *Entertainment Technology - Architecture for Control Networks*. Device Management Protocol.

[SDT] Entertainment Services and Technology Association [https://tsp.esta.org].. ANSI E1.17 - 2010, *Entertainment Technology - Architecture for Control Networks*. Session Data Transport Protocol.

[DDL] Entertainment Services and Technology Association [https://tsp.esta.org].. ANSI E1.17 - 2010,*Entertainment Technology - Architecture for Control Networks*. Device Description Language.

[UDP] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC 768 [http://ietf.org/rfc/rfc0768.txt]. Postel. *User Datagram Protocol*. 1980.

[ESTA-IDs] Entertainment Services and Technology Association [https://tsp.esta.org].. ANSI E1.17 - 2010, *Entertainment Technology - Architecture for Control Networks*. EPI-16 ESTA Registered Names and Identifiers – Format and Procedure for Registration.

[SLPv2] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC 2608 [http://ietf.org/rfc/rfc2608.txt]. Guttman, Perkins, Veizades, and Day. *Service Location Protocol, Version 2*. 1999.

[RLP-UDP] Entertainment Services and Technology Association [https://tsp.esta.org]. ANSI E1.17 - 2010, *Entertainment Technology - Architecture for Control Networks*. EPI 17. Root Layer Protocol Operation on UDP.

[UUID] Internet Engineering Task Force (IETF) [http://ietf.org/]. RFC 4122 [http://ietf.org/rfc/rfc4122.txt]. P. Leach, M. Mealling, and R. Salz. *A Universally Unique IDentifier (UUID) URN Namespace*. July 2005.

[DMX512] Entertainment Services and Technology Association [https://tsp.esta.org]. ANSI E1.11-2004. *USITT DMX512-A - Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories*. 2004.