EtherNet/IP Adaptation of CIP Specification

Release 1.0

June 5, 2001

ControlNet International

and

Open DeviceNet Vendor Association

This page is intentionally left blank

EtherNet/IP Adaptation of CIP Specification
Volume 2

Table of Contents

This page is intentionally left

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 1: Introduction to EtherNet/IP

Contents

1-1 Introduction

EtherNet/IP (Ethernet/Industrial Protocol) is a communication system suitable for use in industrial environments. EtherNet/IP allows industrial devices to exchange time-critical application information. These devices include simple I/O devices such as sensors/actuators, as well as complex control devices such as robots, programmable logic controllers, welders, and process controllers.

EtherNet/IP uses CIP (Control and Information Protocol), the common network, transport and application layers also shared by ControlNet and DeviceNet. EtherNet/IP then makes use of standard Ethernet and TCP/IP technology to transport CIP communications packets. The result is a common, open application layer on top of open and highly popular Ethernet and TCP/IP protocols.

EtherNet/IP provides a producer/consumer model for the exchange of time-critical control data. The producer/consumer model allows the exchange of application information between a sending device (e.g., the producer) and many receiving devices (e.g., the consumers) without the need to send the data multiple times to multiple destinations. For EtherNet/IP, this is accomplished by making use of the CIP network and transport layers along with IP Multicast technology. Many EtherNet/IP devices can receive the same produced piece of application information from a single producing device.

EtherNet/IP makes use of standard IEEE 802.3 technology; there are no non-standard additions that attempt to improve determinism. Rather, EtherNet/IP recommends the use of commercial switch technology, with 100 Mbps bandwidth and full-duplex operation, to provide for more deterministic performance.

NOTE: EtherNet/IP does not require specific implementation or performance requirements due to the broad range of application requirements. However, work is underway to define a standard set of EtherNet/IP benchmarks and metrics by which the performance of devices will be measured. These measurements may become required entries within a product's Electronic Data Sheet. The goal of such benchmarks and metrics will be to help the user determine the suitability of a particular EtherNet/IP device for a specific application.

The figure below illustrates how EtherNet/IP, DeviceNet and ControlNet share the CIP Common layers.

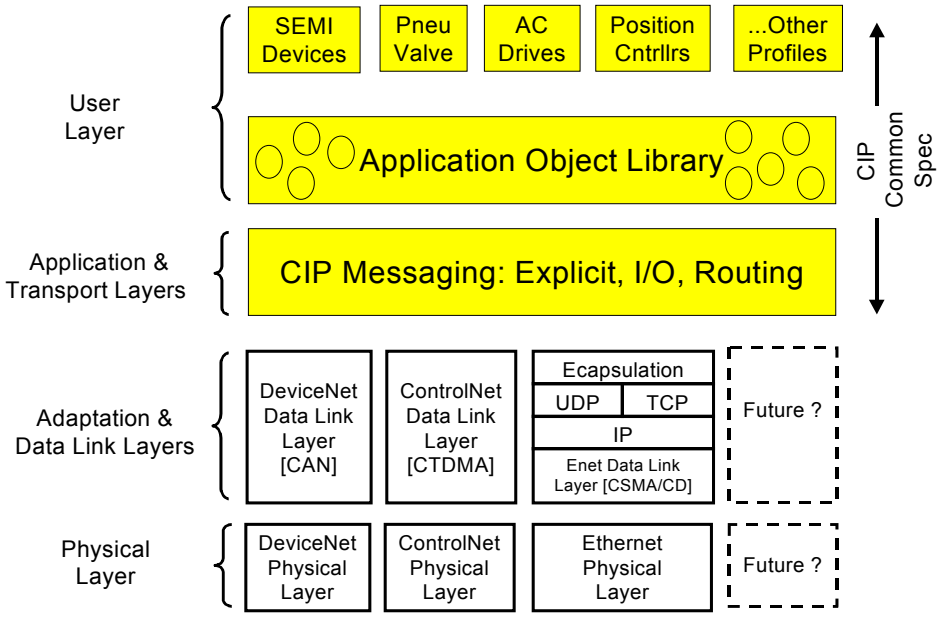


Figure 1-1.1. – CIP Common Overview

1-2 Scope

The EtherNet/IP specification is divided into the following chapters:

Chapter	Title	Description
1	Introduction	This chapter of the specification.
2	Encapsulation Protocol	Specifies the encapsulation protocol that is used to transport CIP packets over TCP/IP networks. The encapsulation protocol specified in this chapter may also be used to encapsulate non-CIP protocols.
3	Mapping of Explicit and I/O Messaging to TCP/IP	Contains EtherNet/IP-specific additions to the CIP Network and Transport layers. Specifies how the encapsulation protocol defined in Chapter 2 is used to transport CIP Network and Transport layer packets over TCP/IP networks.
4	Object Model	Contains EtherNet/IP-specific additions to the CIP object model.
5	Object Library	Supplements the CIP object library with objects specific to EtherNet/IP.
6	Device Profiles	Contains EtherNet/IP-specific additions to the CIP device profile library.
7	Electronic Data Sheets	Specifies additions to the CIP EDS definition required for EtherNet/IP.
8	Physical Layer	Specifies media and physical layer requirements for industrial use.
9	Indicators and Middle Layers	Specifies TCP/IP requirements of EtherNet/IP devices. This chapter also specifies the standard appearance and behaviour of EtherNet/IP diagnostic LEDs.
10	Bridging and Routing	Additions to the CIP routing definition.

This chapter is the Introduction to EtherNet/IP. The following drawing shows the relationship of these chapters to each other and to the CIP Common specification (published separately by ODVA and ControlNet International). Both this specification (volume2) and the CIP Common specification (volume1) are required to completely specify an EtherNet/IP product. The

encapsulation protocol defined in Chapter 2 of this specification is also suitable to encapsulate other industrial protocols, as illustrated in the following drawing. However, the specific details of encapsulating other protocols are not included in this release of the specification.

As can be seen in Figure 1-2.1, the encapsulation protocol in chapter 2 uses a TCP/IP layer to insulate it from the network medium. As such, the encapsulation protocol may be used on any medium that supports TCP/IP. For example, the encapsulation protocol could run on an FDDI or PPP network. Chapter 9 (Indicators and Middle Layers) requires conformance with the RFC that documents how TCP/IP is implemented on a particular network. Furthermore, chapter 8 (Physical Layers) narrows the scope of certified EtherNet/IP implementations to run on either 10 or 100 Mb Ethernet. Specifically, chapter documents two permissible conformance levels of devices: one called "commercial" and the other "industrial". Other conformance levels may be added through modification to this specification.

"Figure 1-2.1 – Document Organization Overview" shows the relationship between the various parts of the EtherNet/IP specification. As shown in the figure, the darker sections (chapters 2-8 and 9) are predominately documented by the CIP Common specification (volume 1). The corresponding chapters of the EtherNet/IP Adaptation of CIP (volume 2) supplements or modifies these chapters of the CIP Common specification in some areas. The lightly shaded sections (chapters 2, 3, 8 and 9) are predominately documented by volume 2. These chapters contain information applicable specifically to EtherNet/IP devices, but not necessarily to those on other CIP networks (for example, DeviceNet or ControlNet).

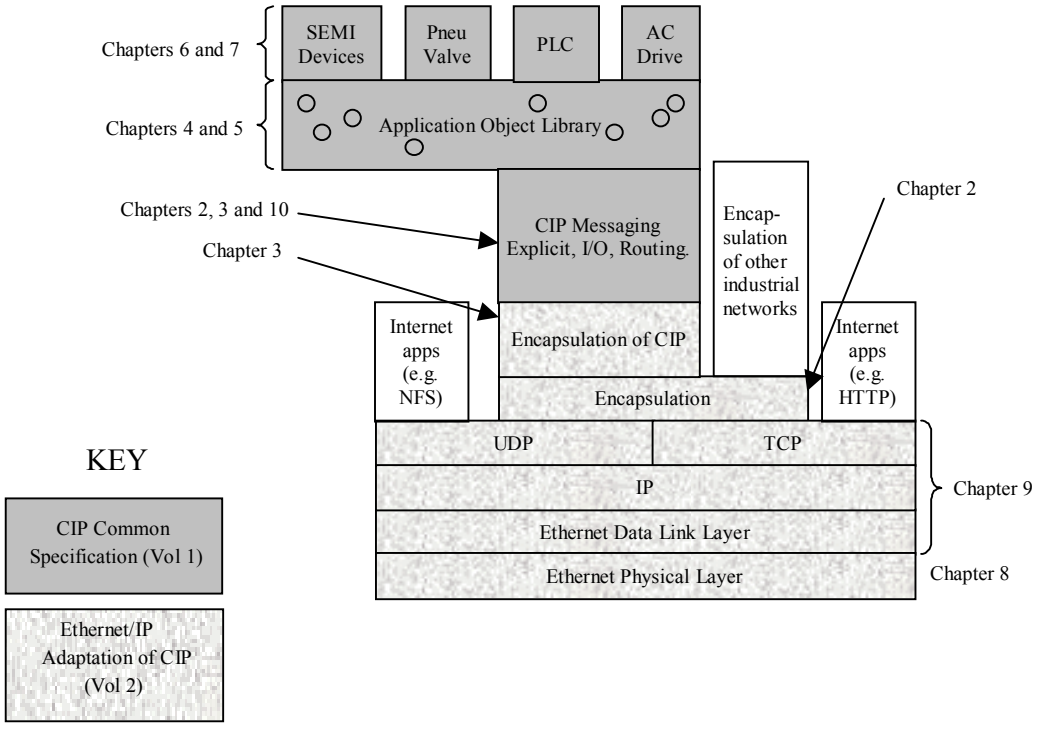


Figure 1-2.1 – Document Organization Overview

1-3 References

1-3.1 Normative References

ISO 7498-1:1984, Information processing systems — Open systems interconnection — Basic reference model

ISO 7498/AD1: 1987, Information processing systems — Open systems interconnection — Connectionless data transmission

ISO 7498-3:1987, Information processing systems — Open systems interconnection — Naming and addressing

ISO/IEC 8886:1992, Information technology — Open systems interconnection — Telecommunications and information exchange between systems — Data link service definition

ISO/IEC 10039:1990, Information technology — Telecommunication and information exchange between systems — Medium access control service definition

ISO/TR 8509:1987, Information processing systems — Open systems interconnection — Service conventions

ISO/IEC 10731:1992, Information technology — Open systems interconnection — Conventions for the definition of OSI services

ISO 8802-2:1989, Information processing systems — Local area networks — Part 2: Logical link control

ISO/IEC 8802-3:1993, Information technology — Local and metropolitan area networks — Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications

ISO/IEC 8802-4:1990, Information processing systems — Local area networks — Part 4: Token - passing bus access method and physical layer specifications

ANSI X3.159-1989, American National Standard for Information Systems — Programming Language C

1-4 Additional Reference Material

"Strategies for Real-time Systems Specification" by D. J. Hatley and I. A. Pirbhai

CEN/CENELEC Internal Regulations Part 3: Rules for the drafting and presentation of European Standards (PNE-Rules) - 1991-09

RFC 768: August 1980, User Datagram Protocol

RFC 791: September 1981, Internet Protocol

RFC 792: September 1981, Internet Control Message Protocol

RFC 793: September 1981, Transmission Control Protocol

RFC 826: November 1982, An Ethernet Address Resolution Protocol

RFC 894: April 1984, A Standard for the Transmission of IP Datagrams over Ethernet Networks

RFC 1035:1987, Domain names - implementation and specification

RFC 1103: June 1989, A Proposed Standard for the Transmission of IP Datagrams over FDDI Networks

RFC 1112: August 1989, Host Extensions for IP Multicasting

RFC 1117:1989, Internet numbers

RFC 1122: October 1989, Requirements for Internet Hosts -- Communication Layers

RFC 1123: October 1989, Requirements for Internet Hosts -- Application and Support

RFC 1127: October 1989, A Perspective on the Host Requirements RFCs

RFC 1171: July 1990, The Point-to-Point Protocol for the Transmission of Multi-Protocol Datagrams Over Point-to-Point Links

RFC 1201: February 1991, Transmitting IP Traffic over ARCNET Networks

RFC 1392: January 1993, Internet Users' Glossary

RFC2236: November 1997, Internet Group Management Protocol, Version 2

1-5 Definitions

For the purposes of this standard, the following definitions apply. Also see CIP Common Specification, Chapter 1 for additional definitions.

1-5.1 broadcast A special type of multicast packet that all nodes on the network are always willing to receive. [Source: RFC1392]

1-5.2 broadcast storm An incorrect packet broadcast onto a network that causes multiple hosts to respond all at once, typically with equally incorrect packets which causes the storm to grow exponentially in severity. [Source: RFC1392]

1-5.3 datagram A self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network. [Source: RFC1392]

1-5.4 encapsulation The technique used by layered protocols in which a layer adds header information to the protocol data unit (PDU) from the layer above. As an example, in Internet terminology, a packet would contain a header from the physical layer, followed by a header from the network layer (IP), followed by a header from the transport layer (TCP), followed by the application protocol data. [Source: RFC1208]

1-5.5 Ethernet A 10-Mb/s standard for LANs, initially developed by Xerox, and later refined by Digital, Intel and Xerox (DIX). All hosts are connected to a coaxial cable where they contend for network access using a Carrier Sense Multiple Access with Collision Detection (CSMA/CD) paradigm. See also: 802.x, Local Area Network, token ring. [Source: RFC1392]

1-5.6 Ethenet/IP Products compliant with this specification as well as the CIP Common specification are known as EtherNet/IP products. EtherNet/IP stands for Ethernet Industrial Protocol. [Source: RFC1392]

1-5.7 frame Single data transfer on a link.

1-5.8 network status indicators Indicators on a node indicating the status of the Physical and Data Link Layers.

1-5.9 port Within the EtherNet/IP specific context, a TCP or UDP port is a transport layer demultiplexing value. Each application has a unique port number associated with it. [Source: RFC1392]. See CIP Common Specification for an additional definition of this term.

1-5.10 redundant media A system using more than one medium to help prevent communication failures.

1-5.11 segment Trunk–cable sections connected via taps with terminators at each end; a segment has no active components and does not include repeaters.

1-5.12 transceiver The physical component within a node that provides transmission and reception of signals onto and off of the medium.

1-6 Abbreviations

For the purposes of this standard, the following abbreviations apply. Also see the CIP Common Specification Chapter 1 for additional abbreviations.

1-6.1 FTP File transfer protocol. An internet application that uses TCP reliable packet transfer to move file between different nodes. (not to be confused with STP/FTP)

1-6.2 LED Light emitting diode

1-6.3 rcv Receive

1-6.4 rx Receive

1-6.5 STP/FTP Shielded twisted pair/foil twisted pair

1-6.6 Tx transmit

1-6.7 UTP Unshielded twisted pair

1-6.8 Xmit transmit

This page is intentionally left blank

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 2: Encapsulation Protocol

Contents

2-1 Introduction

This chapter (chapter 2) of the specification documents the method used to encapsulate industrial protocols on a TCP/IP network. This mechanism can be applied to the CIP industrial protocol or to other networks. Chapter 3 of this specification details the application of this encapsulation protocol to CIP.

With respect to the OSI reference model, this encapsulation protocol inhabits Layer 2 Data Link functions.

2-2 Scope

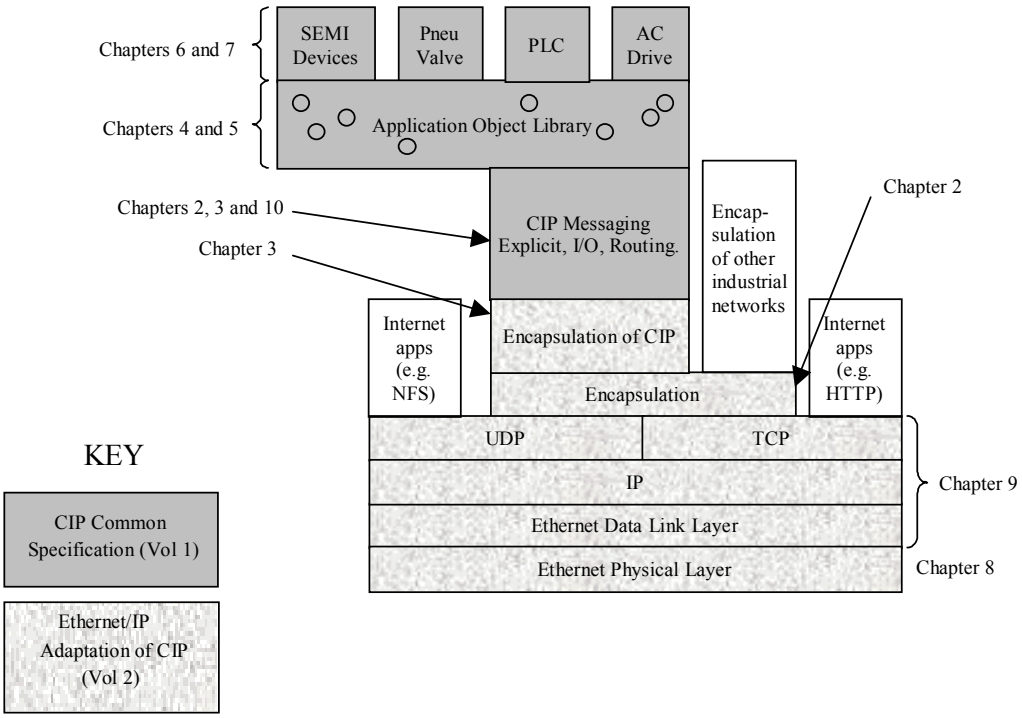


Figure 2-2.1 – Document Organization Overview

2-3 Use of TCP and UDP

The encapsulation protocol defines a reserved TCP port number that shall be supported by all EtherNet/IP devices. All EtherNet/IP devices shall accept at least 2 TCP connections on TCP port number 0xAF12. Once the TCP connection to TCP port number 0xAF12 is established, all data sent through the TCP stream shall be in the format specified in section 2-4.

NOTE: TCP is a stream-based protocol. It is permitted to send almost any length IP packet it chooses. For example, if two back-to-back encapsulated messages where passed to a TCP/IP stack, the TCP/IP stack may choose to put both encapsulated messages in one Ethernet frame. Alternately, it may choose to place half of the first message in the first Ethernet frame and all the rest in the next Ethernet frame. This is shown in Figure 2-3.1 – Usage of TCP to encapsulate two messages.

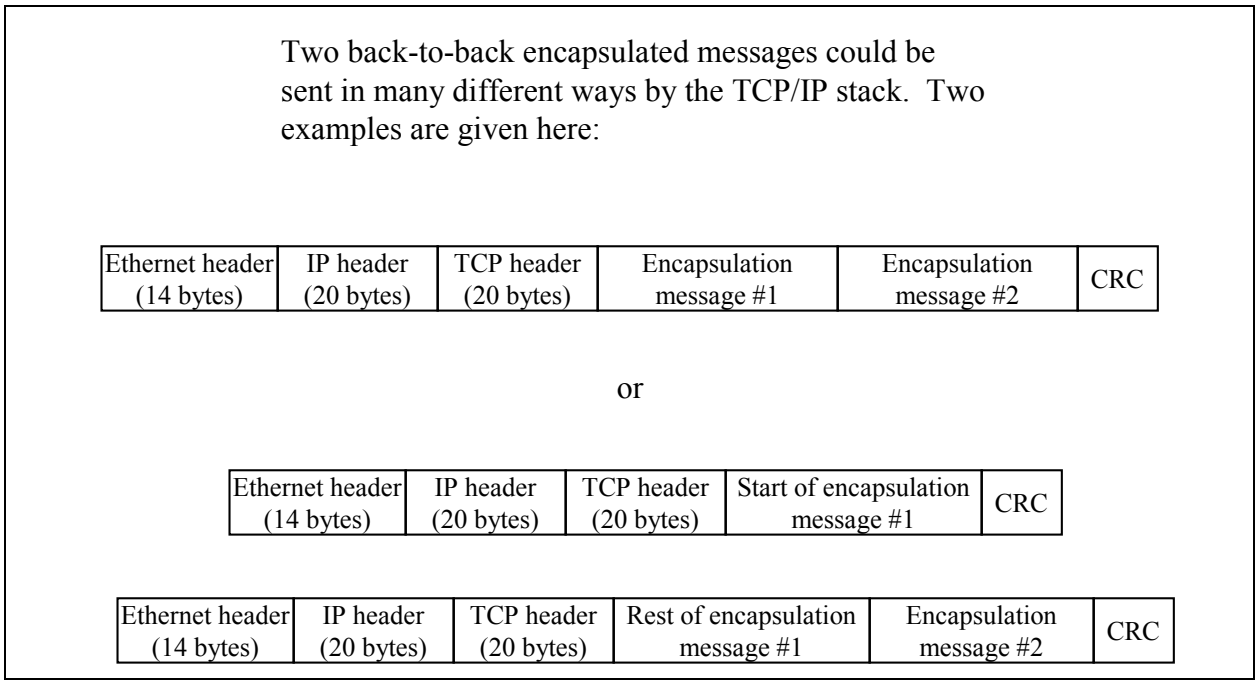


Figure 2-3.1 – Usage of TCP to encapsulate two messages

NOTE: It is not the intention of this specification to document the details of the TCP, UDP and IP transport mechanisms. Many excellent resources including the RFCs referenced throughout this specification should be used to obtain this information.

The encapsulation protocol also defines a reserved UDP port number that shall be supported by all EtherNet/IP devices. All devices shall accept UDP packets on UDP port number 0xAF12. Since UDP, unlike TCP, does not have an ability to reorder packets, whenever UDP is used to send an encapsulated message, the entire message shall be sent in a single UDP packet. Only one encapsulated message shall be present in a single UDP packet destined to UDP port 0xAF12.

Some encapsulated messages shall only be sent via TCP. Other may be sent via either UDP or TCP. See "Table 2-4.2 – Encapsulation commands" for details about which commands are restricted to TCP.

2-4 Encapsulation messages

2-4.1 Encapsulation packet structure

All encapsulation messages, sent via TCP or sent to UDP port 0xAF12, shall be composed of a fixed-length header of 24 bytes followed by an optional data portion. The total encapsulation message length (including header) shall be limited to 65535 bytes. Its structure shall be as follows:

Table 2-4.1 – Encapsulation packet

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	Encapsulation command
	Length	UINT	Length, in bytes, of the data portion of the message, i.e., the number of bytes following the header
	Session handle	UDINT	Session identification (application dependent)
	Status	UDINT	Status code
	Sender Context	ARRAY of octet	Information pertinent only to the sender of an encapsulation command. Length of 8.
	Options	UDINT	Options flags
Command specific data	Encapsulated data	ARRAY of 0 to 65511 USINT	The encapsulation data portion of the message is required only for certain commands

The encapsulation message length shall not override length restrictions imposed by the encapsulated protocol.

NOTE: For example, a CIP UCMM message is still limited to 504 bytes even when encapsulated. See chapter 3 of the CIP Common Specification.

Multi-byte integer fields in the encapsulation messages shall be transmitted in little-endian byte order.

NOTE: This is different from the byte ordering used in standard Internet network protocols, which is big-endian.

Although the header contains no explicit information to distinguish between a request and a reply, this information shall be determined in either of two ways:

- implicitly, by the command and the context in which the message is generated. (For example, in the case of the RegisterSession command, the request is generated by an originator and the target generates the reply);
- explicitly, by the contents of an encapsulated protocol packet in the data part of the message.

2-4.2 Command field

The allocation of command codes shall be as follows:

Table 2-4.2 – Encapsulation commands

Code	Name	Comment
0x0000	NOP	(may be sent only using TCP)
0x0001	Reserved for legacy (RA)	
0x0002 and 0x0003	Reserved for legacy (RA)	
0x0004	ListServices	(may be sent using either UDP or TCP)
0x0005	Reserved for legacy (RA)	

Code	Name	Comment
0x0006 through 0x0062	Reserved for future expansion of this specification (Products compliant with this specification shall not use command codes in this range)	
0x0063	ListIdentity	(may be sent using either UDP or TCP)
0x0064	ListInterfaces	**optional** (may be sent using either UDP or TCP)
0x0065	RegisterSession	(may be sent only using TCP)
0x0066	UnRegisterSession	(may be sent only using TCP)
0x0067 through 0x006E	Reserved for legacy (RA)	
0x006F	SendRRData	(may be sent only using TCP)
0x0070	SendUnitData	(may be sent only using TCP)
0x0071	Reserved for legacy (RA)	
0x0072	IndicateStatus	**optional** (may be sent only using TCP)
0x0073	Cancel	**optional** (may be sent only using TCP)
0x0074 through 0x00C7	Reserved for legacy (RA)	
0x00C8 through 0xFFFF	Reserved for future expansion of this specification (Products compliant with this specification shall not used command codes in this range)	

A device shall accept commands that it does not support without breaking the session or underlying TCP connection. A status code indicating that an unsupported command was received shall be returned to the sender of the message.

NOTE: The establishment of a session is defined in section 2-6. In short, a session makes a TCP/IP connection between originator and target over which encapsulated commands may be sent. Since TCP/IP connections are modeled as a stream of bytes, the encapsulation header is prepended to each encapsulated packet so that the receiving device can know where packets begin and end.

2-4.3 Length field

The length field in the header shall specify the size in bytes of the data portion of the message. The field shall contain zero for messages that contain no data. The total length of a message shall be the sum of the number contained in the length field plus the 24-byte size of the encapsulation header.

The entire encapsulation message shall be read from the TCP/IP connection even if the length is invalid for a particular command or exceeds the target's internal buffers.

NOTE: Failure to read the entire message can result in losing track of the message boundaries in the TCP byte stream.

2-4.4 Session handle

The Session Handle shall be generated by the target and returned to the originator in response to a RegisterSession request. The originator shall insert it in all subsequent encapsulated packets (sent using the commands listed in Table 2-4.2) to that particular target. In the case where the target initiates and sends a command to the originator, the target shall include this field in the request that it sends to the originator.

NOTE: Some commands (i.e., NOP) do not require a session handle even if a session has been established. Whether or not a particular command requires a session is noted in the description of that command.

2-4.5 Status field

The value in the Status field shall indicate whether or not the receiver was able to execute the requested encapsulation command. A value of zero in a reply shall indicate successful execution of the command. In all requests issued by the sender, the Status field shall contain zero. If the receiver receives a request with a non-zero Status field, the request shall be ignored and no reply shall be generated.

NOTE: This field does not reflect errors that are generated by an encapsulated protocol packet contained within the data portion of the message. For example, an error encountered during an end node's processing of a Set Attributes service would be returned via the CIP specified error mechanism (see chapter 3 of the CIP Common specification)..

The status codes shall be as follows:

Table 2-4.3 – Error codes

Status Code	Description
0x0000	Success
0x0001	The sender issued an invalid or unsupported encapsulation command.
0x0002	Insufficient memory resources in the receiver to handle the command. This is not an application error. Instead, it only results if the encapsulation layer cannot obtain memory resources that it needs.
0x0003	Poorly formed or incorrect data in the data portion of the encapsulation message.
0x0004 – 0x0063	Reserved for legacy (RA)
0x0064	An originator used an invalid session handle when sending an encapsulation message to the target.
0x0065	The target received a message of invalid length
0x0066 – 0x0068	Reserved for legacy (RA)
0x0069	Unsupported encapsulation protocol revision.
0x006A – 0xFFFF	Reserved for future expansion (Products compliant with this specification shall not use command codes in this range)

2-4.6 Sender context field

The sender of the command shall assign the value in the Sender Context field of the header. The receiver shall return this value without modification in its reply. Commands with no expected reply may ignore this field.

NOTE: The sender of a command may place any value in this field. It could be used to match requests with their associated replies.

2-4.7 Options field

The sender of an encapsulated packet shall set the options field to zero. The receiver of an encapsulated packet shall verify that the option field is zero. The receiver shall discard encapsulated packets with a non-zero option field.

NOTE: The intent of this field is to provide bits that modify the meaning of the various encapsulation commands. No particular use for this field has not yet been specified.

2-4.8 Command specific data field

NOTE: The structure of the command specific data field depends on the command code. To organize their command specific data field, most commands use either or both of the following two methods:

1) use a fixed structure
2) use the common packet format (described in section 2-7)

The common packet format allows commands to structure their command specific data field in an extensible way.

2-5 Command descriptions

2-5.1 NOP

Either an originator or a target may send a NOP command. No reply shall be generated by this command. The data portion of the command shall be from 0 to 65511 bytes long. The receiver shall ignore any data that is contained in the message.

NOTE: A NOP provides a way for either an originator or target to determine if the TCP connection is still open.

The NOP encapsulation header shall be as follows:

Table 2-5.1 – NOP header values

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	NOP (0x00)
	Length	UINT	Length of data portion (from 0 to 65511)
	Session handle	UDINT	This field is ignored since the NOP command does not generate a reply.
	Status	UDINT	shall be 0
	Sender Context	ARRAY of octet	This field is ignored since the NOP command does not generate a reply. Length of 8.
	Options	UDINT	shall be 0
Command specific data	Unused data	ARRAY of USINT	unused data

2-5.2 ListIdentity

2-5.2.1 General

A connection originator may use the ListIdentity command to locate and identify potential targets. This command shall be sent as a broadcast message using UDP and does not require that a session be established.

2-5.2.2 Request

The ListIdentity request shall be as shown below:

Table 2-5.2 – ListIdentity request

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	ListIdentity (0x63)
	Length	UINT	0
	Session handle	UDINT	This field is ignored since a session need not be established before sending the ListIdentity request.
	Status	UDINT	0
	Sender Context	ARRAY of octet	0
	Options	UDINT	0

2-5.2.3 Reply

One reply item is defined for this command, Target Identity, with item type code 0x0C. This item shall be supported (returned) by all CIP capable devices.

Each receiver of the List Identity command shall reply with a standard encapsulation header and data, as shown below. The data portion of the message shall provide the information on the targets identity. The reply shall be sent to the IP address from which the broadcast request was received.

Table 2-5.3 – ListIdentity reply

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	List Identity (0x63)
	Length	UINT	0
	Session handle	UDINT	Ignored
	Status	UDINT	0
	Sender Context	ARRAY of octet	0
	Options	UDINT	0
Command specific data	Item Count	UINT	Number of target items to follow
	ListIdentity Items	STRUCT of	Interface Information
		UINT	Item Type Code
		UINT	Item Length
		ARRAY of byte	Item Data

The data portion of the message shall be the Common Packet Format that contains a 2-byte item count followed by an array of items providing the target identity.

At a minimum, the CIP Identity item shall be returned and has the format as defined in Table 2-5.4. Part of this item definition follows the Get Attribute All service response definition of the Identity Object (data returned based on instance one of this object), and may adopt new members if and when new members are added to that service response. Unlike most fields in the Common Packet Format, the Socket Address field shall be sent in big endian order.

Table 2-5.4 – CIP Identity item

Parameter Name	Data Type	Description
Item Type Code	UINT	Code indicating item type of CIP Identity (0x0C)
Item Length	UINT	Number of bytes in item which follow (length varies depending on Product Name string)
Encapsulation Protocol Version	UINT	Encapsulation Protocol Version supported (also returned with Register Sesstion reply).
Socket Address	STRUCT of	Socket Address (see section 2-7.3.2)
	INT	sin_family (**big-endian**)
	UINT	sin_port (**big-endian**)
	UDINT	sin_addr (**big-endian**)
	ARRAY of USINT	sin_zero (length of 8) (**big-endian**)
Vendor ID[1]	UINT	Device manufacturers Vendor ID
Device Type[1]	UINT	Device Type of product
Product Code[1]	UINT	Product Code assigned with respect to device type
Revision[1]	USINT[2]	Device revision
Status[1]	WORD	Current status of device
Serial Number[1]	UDINT	Serial number of device
Product Name[1]	SHORT_STRING	Human readable description of device
State[1]	USINT	Current state of device

[1] These parameters are further defined by the corresponding instance attribute of the Identity Object. (see the CIP Common specification, chapter 5, Object Library)

2-5.3 ListInterfaces

2-5.3.1 General

The optional List Interfaces command shall be used by a connection originator to identify potential non-CIP communication interfaces associated with the target. A session need not be established to send this command.

2-5.3.2 Request

The ListInterfaces request shall be as shown below.

Table 2-5.5 – ListInterfaces request

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	List Interfaces (0x64)
	Length	UINT	0
	Session handle	UDINT	Ignored
	Status	UDINT	0
	Sender Context	ARRAY of octet	0
	Options	UDINT	0

2-5.3.3 Reply

If supported, the receiver of a ListInterfaces request command shall reply with a standard encapsulation header and data, as shown below. The data portion of the message is structured as a Common Packet Format and shall provide information on the non-CIP communication interfaces associated with the target.

Table 2-5.6 – ListInterfaces reply

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	List Interfaces (0x64)
	Length	UINT	0
	Session handle	UDINT	Ignored
	Status	UDINT	0
	Sender Context	ARRAY of octet	0
	Options	UDINT	0
Command specific data	Item Count	UINT	Number of target items to follow
	Target Items	STRUCT of	Interface Information
		UINT	Item Type Code
		UINT	Item Length
		ARRAY of byte	Item Data

The data portion of the message shall be the Common Packet Format, which contains a 2-byte item count followed by an array of items providing interface information. There are no publicly defined items returned with this reply. The vendor-specific item(s) which is/are returned shall, at a minimum, return a 32 bit Interface Handle which is used by other encapsulation commands, for example, the SendRRData command.

2-5.4 RegisterSession

2-5.4.1 General

An originator shall send a RegisterSession command to a target to initiate a session.

NOTE: See section 2-6, for detailed information on establishing and maintaining a session.

2-5.4.2 Request

The RegisterSession request shall be as follows:

Table 2-5.7 – RegisterSession request

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	RegisterSession (0x65)
	Length	UINT	4 bytes
	Session handle	UDINT	0
	Status	UDINT	0
	Sender Context	ARRAY of octet	Any sender context. Length of 8.
	Options	UDINT	0
Command specific data	Protocol version	UINT	Requested protocol version shall be set to 1.
	Options flags	UINT	Session options shall be set to 0 Bits 0-7 are reserved for legacy (RA) Bits 8-15 are reserved for future expansion **NOTE**: This field is not the same as the option flags in the encapsulation header.

2-5.4.3 Reply

The target shall send a RegisterSession reply to indicate that it has registered the originator. The reply shall have the same format as the request as shown below:

Table 2-5.8 – RegisterSession reply

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	RegisterSession (0x65)
	Length	UINT	4 bytes
	Session handle	UDINT	handle returned by target
	Status	UDINT	0
	Sender Context	ARRAY of octet	context preserved from the corresponding RegisterSession request. Length of 8.
	Options	UDINT	0
Command specific data	Protocol version	UINT	Requested protocol version shall be set to 1.
	Options flags	UINT	Session options shall be set to 0 Bits 0-7 are reserved for legacy (RA) Bits 8-15 are reserved for future expansion **NOTE**: This field is not the same as the option flags in the encapsulation header.

The Session Handle field of the header shall contain a target-generated identifier that the originator shall save and insert in the Session Handle field of the header for all subsequent requests to that target. This field shall be valid only if the Status field is zero (0).

The Sender Context field of the header shall contain the same values present in the original sender request. If the originator has been registered with the target, the Status field shall be zero (0). If the target was unable to register, the Status field shall be set to 0x69 (unsupported encapsulation protocol revision).

The Protocol Version field shall equal the requested version if the originator was successfully registered. If the target does not support the requested version of the protocol,

- the session shall not be created;
- the Status field shall be set to unsupported encapsulation protocol 0x69;
- the target shall return the highest supported version in the Protocol Version field.

If all requested options are supported, the Options field shall return the originator's value. This value shall be zero.

2-5.5 UnRegisterSession

Either an originator or a target may send this command to terminate the session. The receiver shall initiate a close of the underlying TCP/IP connection when it receives this command. The session shall also be terminated when the transport connection between the originator and target is terminated. The receiver shall perform any other associated cleanup required on its end. There shall be no reply to this command.

The UnregisterSession command format shall be as follows:

Table 2-5.9 – UnregisterSession command

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	UnRegisterSession (0x66)
	Length	UINT	0 bytes
	Session handle	UDINT	handle from RegisterSession reply
	Status	UDINT	shall be 0
	Sender Context	ARRAY of octet	Any sender context. Length of 8.
	Options	UDINT	shall be 0

The Session Handle shall be set to the value obtained by the original RegisterSession reply. Once the client has sent this command, it shall no longer use the handle. .

NOTE: See section 2-6.3 for more detail about terminating a session.

2-5.6 ListServices

2-5.6.1 General

The ListServices command shall determine which encapsulation service classes the target device supports.

NOTE: Each service class has a unique type code, and an optional ASCII name.

2-5.6.2 Request

The ListServices header shall be as follows:

Table 2-5.10 – ListServices request

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	ListServices (0x04)
	Length	UINT	0 bytes
	Session handle	UDINT	ignored
	Status	UDINT	0
	Sender Context	ARRAY of octet	Any sender context. Length of 8.
	Options	UDINT	0

2-5.6.3 Reply

The receiver shall reply with a standard encapsulation message, consisting of the header and data, as shown below. The data portion of the message shall provide the information on the services supported.

Table 2-5.11 – ListServices reply

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	ListServices (0x04)
	Length	UINT	0 bytes
	Session handle	UDINT	ignored
	Status	UDINT	0
	Sender Context	ARRAY of octet	Preserved from the corresponding ListServices request. Length of 8.
	Options	UDINT	0
Command specific data	Item Count	UINT	Number of items to follow
	Target Items	STRUCT of	Interface Information
		UINT	Item Type Code
		UINT	Item Length
		UINT	Version of encapsulated protocol shall be set to 1
		UINT	Capability flags
		ARRAY of 16 USINT	Name of service

The Type Code shall identify the service class as follows:

One service class is defined, with type code 0x100 and name "Communications". This service class shall indicate that the device supports encapsulation of CIP packets. All devices that support encapsulating CIP shall support the ListServices request and Communications service class.

NOTE: See section 2-7 for a description of items and a list of all reserved item codes.

The Version field shall indicate the version of the service supported by the target to help maintain compatibility between applications.

Each service shall have a different set of capability flags. Unused flags shall be set to zero.

The Capability Flags, defined for the Communications service, shall be as follows:

Table 2-5.12 – Capability flags

Flag Value	Description
Bits 0 – 4	reserved for legacy (RA)
Bit 5	If the device supports CIP packet encapsulation via TCP this bit shall be set (=1); otherwise, it shall be clear (=0)
Bits 6 – 7	reserved for legacy (RA)
Bit 8	Supports CIP Class 0 or 1 UDP-based connections
Bits 9 – 15	Reserved for future expansion

The Name field shall allow up to a 16-byte, NULL-terminated ASCII string for descriptive purposes only. The 16-byte limit shall include the NULL character.

2-5.7 SendRRData

2-5.7.1 General

A SendRRData command shall transfer an encapsulated request/reply packet between the originator and target, where the originator initiates the command. The actual request/reply packets shall be encapsulated in the data portion of the message and shall be the responsibility of the target and originator.

NOTE: When used to encapsulate the CIP, the SendRRData request and response are used to send encapsulated UCMM messages (unconnected messages). See chapter 3 for more detail.

2-5.7.2 Request

The SendRRData header shall be as follows:

Table 2-5.13 – SendRRData request

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	SendRRData (0x6F)
	Length	UINT	Length of data portion
	Session handle	UDINT	Handle returned by RegisterSession
	Status	UDINT	0
	Sender Context	ARRAY of octet	Any sender context. Length of 8.
	Options	UDINT	0
Command specific data	Interface handle	UDINT	shall be 0 for CIP
	Timeout	UINT	operation timeout
	Encapsulated packet	ARRAY of octet	see Common Packet Format specification in section 2-7)

The Interface handle shall identify the Communications Interface to which the request is directed. This handle shall be 0 for encapsulating CIP packets.

The target shall abort the requested operation after the timeout expires. When the “timeout” field is in the range 1 to 65535, the timeout shall be set to this number of seconds. When the “timeout” field is set to 0, the encapsulation protocol shall not have its own timeout. Instead, it shall rely on the timeout mechanism of the encapsulated protocol.

NOTE: When used to encapsulate CIP packets, the timeout field is usually set to 0 since CIP provides its own timeout mechanism for connected messages.

The encapsulated protocol packet shall be encoded in the Common Packet Format as shown in section 2-7.

2-5.7.3 Reply

The SendRRData reply, as shown below, shall contain data in response to the SendRRData request. The reply to the original encapsulated protocol request shall be contained in the data portion of the SendRRData reply.

Table 2-5.14 – SendRRData reply

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	SendRRData (0x6F)
	Length	UINT	length of data structure
	Session handle	UDINT	handle returned by RegisterSession
	Status	UDINT	0
	Sender Context	ARRAY of octet	Preserved from the corresponding SendRRData request. Length of 8.
	Options	UDINT	0
Command specific data	Interface handle	UDINT	shall be 0 for CIP
	Timeout	UINT	operation timeout (not used)
	Encapsulated packet	ARRAY of octet	see Common Packet Format specification in section 2-7)

The format of the data portion of the reply message shall be the same as that of the SendRRData request message.

NOTE: Since the request and reply share a common format, the reply message contains a timeout field; however, it is not used.

2-5.8 SendUnitData

The SendUnitData command shall send encapsulated connected messages. This command may be used when the encapsulated protocol has its own underlying end-to-end transport mechanism. A reply shall not be returned. The SendUnitData command may be sent by either end of the TCP connection.

NOTE: When used to encapsulate the CIP, the SendUnitData command is used to send CIP connected data in both the O⇒T and T⇒O directions.

The format of the SendUnitData command shall be as follows:

Table 2-5.15 – SendUnitData command

Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	SendUnitData (0x70)
	Length	UINT	Length of data portion
	Session handle	UDINT	Handle returned by RegisterSession
	Status	UDINT	0
	Sender Context	ARRAY of octet	Any sender context. Length of 8.
	Options	UDINT	0
Command specific data	Interface handle	UDINT	shall be 0
	Timeout	UINT	shall be 0
	Encapsulated packet	ARRAY of octet	see Common Packet Format specification in section 2-7)

Interface handle and Timeout shall be set the zero. The timeout field is not used since no reply is generated upon receipt of a SendUnitData command.

2-6 Session management

2-6.1 Phases of a TCP encapsulation session

An encapsulation session shall have three phases:

- establishing a session;
- maintaining a session;
- closing a session.

2-6.2 Establishing a session

Session establishment shall proceed according to the following steps:

- The originator shall open a TCP/IP connection to the target, using the reserved TCP port number (0xAF12), or if specified, the TCP port number from the connection path (the means to specify an alternate TCP port number is described in chapter 3);
- The originator shall send a RegisterSession command to the target (see section 2-5.4 for a description of the RegisterSession command);
- The target shall check the protocol version in the command message to verify it supports the same protocol version as the originator. If not, the target shall return a RegisterSession with an appropriate Status field along with the highest supported protocol version;
- The target shall assign a new (unique) Session ID and shall send a RegisterSession reply to the originator.

2-6.3 Terminating a session

Either the originator or the target may terminate the session. Sessions shall be terminated in either of two ways:

- The originator or target shall close the underlying TCP connection. The corresponding target or originator shall detect the loss of the TCP connection, and shall close its side of the connection;
- The originator or target shall send an UnRegisterSession command (see section 2-5.4 for a description of the UnregisterSession command) and shall wait to detect the closing of the TCP connection. The corresponding target or originator shall then close its side of the TCP connection. The sender of the UnRegisterSession shall detect the loss of the TCP connection, then it shall close its side of the connection.

NOTE: The second method is preferred since it results in more timely clean up of the TCP connection.

2-6.4 Maintaining a session

Once a session is established, it shall remain established until one of the following occurs:

- the originator or target closes the TCP connection;
- the originator or target issues the UnRegisterSession command;
- the TCP connection is broken.

2-6.5 TCP behavior (informative)

TCP is a reliable, connection-oriented protocol. If a process at either end of a connection closes its end of the connection, the TCP at the other end is notified immediately. If a message from one process to the other can not be delivered in a reasonable amount of time, the connection is assumed to be broken and an error is returned to the sender on all subsequent sends and receives on the connection.

If an originator process detects that a target has closed its end of a connection or that a connection is broken, it assumes the session with the target is broken and closes its connection to the target. A new session is then established as described above in order to resume communications with the target.

Although an originator process is notified when the other end of a connection has been closed, a broken connection can only be detected when a process actually attempts to send a message over the connection. In most cases, the originator process sends messages to targets frequently enough that a crash of a target machine is detected in a timely manner. Likewise, targets send messages back to originators frequently enough that terminated originator processes and originator machine crashes are detected quickly. However, it is possible that an originator or target may not have any messages to send on a connection for a relatively long period of time.

The TCP protocol supports keep-alive processing. An application can ask TCP to make sure the connection remains working during periods when the application does not have any messages to send. If this feature is enabled, when the connection has been idle for some period of time, TCP will send a keep-alive message to its peer at the other end of the connection. If TCP sends several keep-alive messages and does not receive a reply, TCP assumes the

connection has broken and the application is notified just as if it had sent an actual message that timed out.

Most implementations of TCP/IP retry/timeout processing do not declare a failure on a connection until it has remained unusable for several minutes. This is a feature of the TCP protocol on the originator host; turning keep alives does not modify it.

2-7 Common packet format

2-7.1 General

The common packet format shall consist of an item count, followed by an address item, then a data item (in that order) as shown below. Additional optional items may follow.

NOTE: The common packet format defines a standard format for protocol packets that are transported with the encapsulation protocol. The common packet format is a general-purpose mechanism designed to accommodate future packet or address types.

Table 2-7.1 – Common packet format

Field Name	Data Type	Description
Item count	UINT	Number of items to follow (shall be at least 2)
Address item	Item Struct (see below)	Addressing information for encapsulated packet
Data item	Item Struct (see below)	The encapsulated data packet
Optional additional items		

The address and data item structure shall be as follows:

Table 2-7.2 – Data and Address item format

Field Name	Data Type	Description
Type ID	UINT	Type of item encapsulated
Length	UINT	Length in bytes of data to follow
Data	Variable	The data (if length >0)

Table 2-7.3 – Item ID numbers

Item ID number	Item type	Description
0x0000	address	Null (used for UCMM messages) Indicates that encapsulation routing is NOT needed. Target is either local (ethernet) or routing info is in a data Item.
0x0001 – 0x000B		Reserved for legacy (RA)
0x000C		ListIdentity response
0x000D – 0x0083		Reserved for legacy (RA)
0x0084 – 0x0090		Reserved for future expansion of this specification (Products compliant with this specification shall not use command codes in this range)
0x0091		Reserved for legacy (RA)
0x0092 – 0x00A0		Reserved for future expansion of this specification (Products compliant with this specification shall not use command codes in this range)

Item ID number	Item type	Description
0xA1	address	Connection-based (used for connected messages)
0x00A2 – 0x00A4		Reserved for legacy (RA)
0x00A5 – 0x00B0		Reserved for future expansion of this specification (Products compliant with this specification shall not use command codes in this range)
0x00B1	data	Connected Transport packet
0x00B2	data	Unconnected message
0x00B3 – 0x00FF		Reserved for future expansion of this specification (Products compliant with this specification shall not use command codes in this range)
0x0100		ListServices response
0x0101 – 0x010F		Reserved for legacy (RA)
0x0110 – 0x7FFF		Reserved for future expansion of this specification (Products compliant with this specification shall not use command codes in this range)
0x8000	data	Sockaddr Info, originator-to-target
0x8001	data	Sockaddr Info, target-to-originator
0x8002		Sequenced Address iteme
0x8003 – 0xFFFF		Reserved for future expansion of this specification (Products compliant with this specification shall not use command codes in this range)

2-7.2 Address items

2-7.2.1 Null address item

The null address item shall contain only the type id and the length as shown below. The length shall be zero. No data shall follow the length. Since the null address item contains no routing information, it shall be used when the protocol packet itself contains any necessary routing information. The null address item shall be used for Unconnected Messages.

Table 2-7.4 – Null address item

Field Name	Data Type	Field Value
Type ID	UINT	0
Length	UINT	0

2-7.2.2 Connected address item

This address item shall be used when the encapsulated protocol is connection-oriented. The data shall contain a connection identifier.

NOTE: Connection identifiers are exchanged in the Forward_Open service of the Connection Manager.

Table 2-7.5 – Connected address item

Field Name	Data Type	Field Value
Type ID	UINT	0xA1
Length	UINT	4
Data	UDINT	Connection Identifier

2-7.2.3 Sequenced address item

This address item shall be used for CIP transport class 0 and class 1 connected data. The data shall contain a connection identifier and a sequence number.

Table 2-7.6 – Sequenced address item

Field Name	Data Type	Field Value
Type ID	UINT	0x8002
Length	UINT	8
Data	UDINT	Connection Identifier
	UDINT	Sequence Number

2-7.3 Data items

2-7.3.1 Unconnected data item

The data item that encapsulates an unconnected message shall be as follows:

Table 2-7.7 – Unconnected data item

Field Name	Data Type	Field Value
Type ID	UINT	0xB2
Length	UINT	Length, in bytes, of the unconnected message
Data	Variable	The unconnected message

NOTE: The format of the "data" field is dependent on the encapsulated protocol. When used to encapsulate CIP, the format of the "data" field is that of a Message Router request or Message Router reply. See chapter 3 of this specification for details of the encapsulation of UCMM messages. See chapter 2 of the CIP Common specification for the format of the Message Router request and reply packets.

The context field in the encapsulation header shall be used for unconnected request/reply matching.

2-7.3.2 Connected data item

The data item that encapsulates a connected transport packet shall be as follows:

Table 2-7.8 – Connected data item

Field Name	Data Type	Field Value
Type ID	UINT	0xB1
Length	UINT	Length, in bytes, of the transport packet
Data	Variable	The transport packet

NOTE: The format of the "data" field is dependent on the encapsulated protocol. When used to encapsulate CIP, the format of the "data" field is that of connected packet. See chapter 3 of this specification for details of the encapsulation of connected packets. See chapter 3 of the CIP Common specification for the format of connected packets.

2-7.3.3 Sockaddr info item

The Sockaddr Info items shall be used to encapsulate socket address information necessary to send datagrams (the connected data) between the target and originator. There are separate items for originator-to-target and target-to-originator socket information.

The Sockaddr Info items shall have the following structure:

Table 2-7.9 – Sockaddr item

Field Name	Data Type	Field Value
Type ID	UINT	0x8000 for O⇒T, 0x8001 for T⇒O
Length	UINT	16 (bytes)
sin_family	INT	shall be AF_INET = 2. This field shall be sent in big endian order.
sin_port	UINT	shall be set to the TCP or UDP port on which packets for this CIP connection will be sent. This field shall be sent in big endian order.
sin_addr	UDINT	shall be set to the IP address to which packet for this CIP connection will be sent. This field shall be sent in big endian order.
sin_zero	ARRAY of USINT	shall be 0. This field shall be sent in big endian order. Length of 8.

NOTE: The structure of the Sockaddr item has been patterned after the sockaddr_in structure from the Winsock specification, version 1.1.

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 3: Mapping of Explicit and I/O Messaging to TCP/IP

Contents

3-1 Introduction

This chapter (chapter 3) of the EtherNet/IP specification describes the application of the encapsulation in chapter 2 to the control and information protocol (CIP). Specifically, this chapter documents the encapsulation of the UCMM and connected packets; extends the format of the path to include IP addresses; and limits which CIP transport parameters can be used in combination.

3-2 Scope

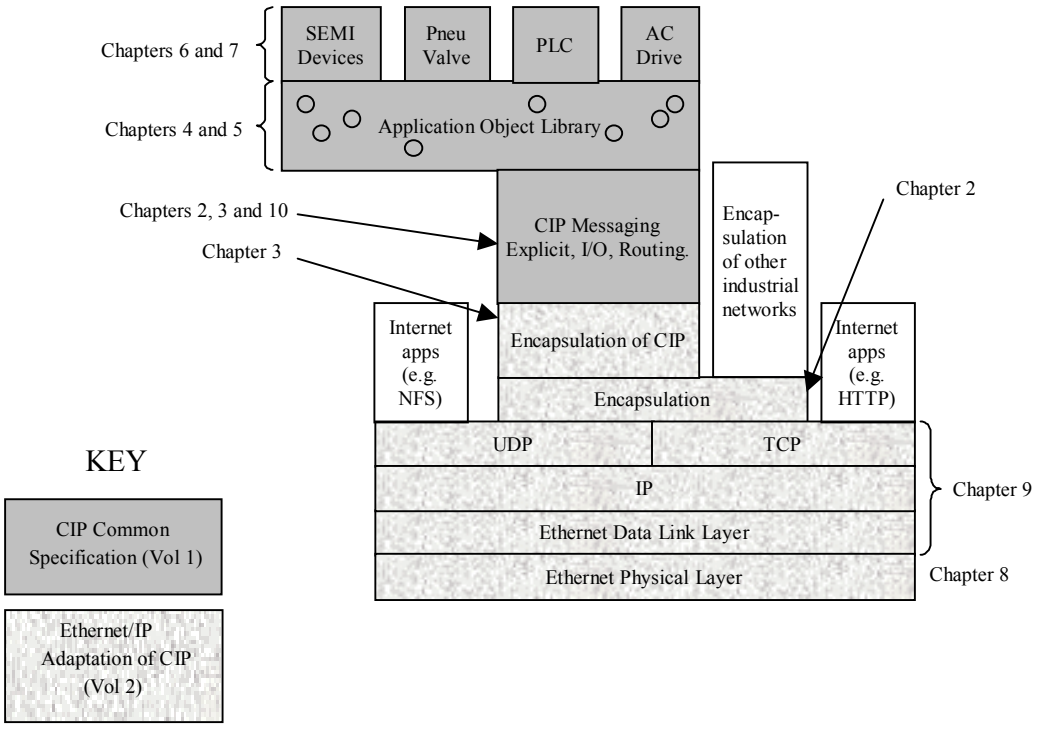


Figure 3-2.1 – Document Organization Overview

3-3 CIP packets over TCP/IP

When the path of a CIP packet traverses an Ethernet-TCP/IP network, the encapsulated packet shall be transmitted using the TCP/IP protocol suite and the encapsulation protocol defined in chapter 2.

3-3.1 Unconnected messages

UCMM packets shall be transmitted over a TCP/IP connection, using the encapsulation protocol defined in chapter 2. For example, an encapsulated UCMM request shall be formatted as shown in Table 3-3.1.

Table 3-3.1 – UCMM request

Structure	Field Name		Data Type	Field Value
Encapsulation header	Command		UINT	SendRRData (0x6F)
	Length		UINT	Length of command specific data portion
	Session handle		UDINT	Handle returned by RegisterSession
	Status		UDINT	0
	Sender Context		ARRAY of 8 USHORT	any sender context
	Options		UDINT	0
Command specific data	Interface handle		UDINT	shall be 0 for CIP
	Timeout		UINT	operation timeout
	Encapsulated packet (in the common packet format)	Item count	UINT	This field shall be 2 since one address item and one data item are used.
		Address Type ID	UINT	This field shall be 0 to indicate a UCMM message.
		Address Length	UINT	This field shall be 0 since UCMM messages use the NULL address item.
		Data Type ID	UINT	This field shall be 0x00B2 to encapsulate the UCMM
		Data Length	UINT	Length of the next field in bytes (length of the MR request packet)
		MR request packet	ARRAY of USINT	This field contains a CIP Message Router request packet as defined in chapter 2 of the CIP Common specification.

Likewise, the UCMM reply shall be formatted as shown in Table 3-3.2.

Table 3-3.2 – UCMM reply

Structure	Field Name		Data Type	Field Value
Encapsulation header	Command		UINT	SendRRData (0x6F)
	Length		UINT	Length of command specific data portion
	Session handle		UDINT	Handle returned by RegisterSession
	Status		UDINT	0
	Sender Context		ARRAY of 8 USHORT	copied from the corresponding UCMM request
	Options		UDINT	0
Command specific data	Interface handle		UDINT	shall be 0 for CIP
	Timeout		UINT	not used for reply
	Encapsulated packet (in the common packet format)	Item count	UINT	This field shall be 2 since one address item and one data item are used.
		Address Type ID	UINT	This field shall be 0 to indicate a UCMM message.
		Address Length	UINT	This field shall be 0 since UCMM messages use the NULL address item.
		Data Type ID	UINT	This field shall be 0x00B2 to encapsulate the UCMM
		Data Length	UINT	Length of the next field in bytes (length of the MR response packet)
		MR response packet	ARRAY of USINT	This field contains a CIP Message Router reply packet as defined in chapter 2 of the CIP Common specification.

3-3.2 CIP transport class 0 and class 1 connections

NOTE: Please see the CIP Common specification for the definition and usage of CIP transport class 0 and class 1 connections.

3-3.2.1 CIP transport class 0 and class 1 packets

Packets for transport class 0 and class 1 connections shall be transmitted using UDP and the Common Packet Format defined in chapter 2 of the CIP on EtherNet/IP specification. Packets for multicast connections shall be transmitted using IP multicast.

3-3.2.2 Behavior of class 0 and class 1 connections (informative)

Since Ethernet does not have a mechanism for sending scheduled data, several important aspects of class 0 and class 1 behavior are noted as follows:

On Ethernet, it is possible for a CIP transport class 0 or class 1 connected data packet to be lost, for example due to excessive collisions. By definition, class 0 and class 1 connections do not guarantee the delivery of every packet. Rather, producers simply send data at the specified rate (the API). If a packet is lost on a class 0 or class 1 connection, the consumer receives the next packet from the producer.

The degree to which lost packets can be tolerated is application-specific. Ethernet is not suitable for those applications that cannot tolerate any lost packets.

For CIP transport class 1 connections, the consuming CIP transport can detect packet loss by examining the CIP sequence number in the class 1 packet. For class 0 connections, it is not possible for the application to know that a specific packet has been lost since class 0 does not use a sequence number.

The connection timeout mechanism provides feedback to the application when too many packets are lost. The connection timeout is determined by the Requested Packet Interval (RPI) and by the Connection Timeout Multiplier. If a packet is not received in the time specified by the RPI times the Connection Timeout Multiplier, the connection is broken. For example, if the RPI is 50 ms and the Connection Timeout Multiplier is 4, then the connection will time out if a fresh packet is not received in 200 ms (the equivalent of 4 packets being lost). Receipt of older packets (those with lower CIP sequence numbers) will not sustain the CIP connection.

The degree of packet loss for any particular connection will be dependent upon many factors related to the user's Ethernet network configuration. It is beyond the scope of this specification to address this in further detail.

3-3.3 CIP Transport class 2 and class 3 connections

CIP transport class 2 and class 3 connections shall be transmitted over a TCP connection using the encapsulation protocol defined in chapter 2.

Multiple CIP connections may be sent over a single TCP connection. An implementation need not support a specific number of CIP connections per TCP connection. An implementation may impose an upper bound if it chooses.

Because of the full-duplex nature of TCP, the CIP originator to target (O⇒T) and CIP target to originator (T⇒O) link connections shall use the same TCP connection. However if a target subsequently originates a CIP connection, then it shall be considered an originator, and a different TCP connection shall be used.

NOTE: This standard defines no requirements for management of the TCP connection, such as inactivity timeouts, or closing the TCP connection when all native connections are closed. However, implementations are free to implement these.

3-3.4 CIP Transport classes 4 through 6

The encapsulation protocol described in chapter 2 shall not be used to encapsulate CIP transport classes 4, 5 and 6.

3-4 Connection Manager object

3-4.1 Connection parameters

NOTE: This section documents the Connection Manager parameters that have requirements specific to the TCP/IP encapsulation. Connection Manager parameters are fully described in chapter 3 of the CIP common specification.

3-4.2 Connection type

The CIP connection type shall be `NULL, MULTICAST,` or `POINT2POINT`. The `MULTICAST` connection type shall be supported only for CIP transport class 0 and class 1 connections.

3-4.3 Priority

The CIP priority shall be `LOW`, or `HIGH`, or `SCHEDULED`. At present, `SCHEDULED` priority shall be treated no differently than `HIGH` priority.

NOTE: SCHEDULED priority for Ethernet-TCP/IP connections may be further defined in the future.

3-4.4 Trigger Type

The CIP trigger type shall be `CYCLIC, CHANGE_OF_STATE, or APPLICATION`. CIP transport class 0 and class 1 connections that use `CHANGE_OF_STATE` triggering shall use the Production Inhibit Time segment (see the CIP Common specification).

3-4.5 Connection size

The CIP connection size shall be no larger than 65511 bytes.

NOTE: The Forward_Open request limits the connection size to 511 bytes; however, the optional Ex_Forward_Open allows larger connection sizes.

3-4.6 Connection request time-out

To reliably establish a CIP connection that extends onto a TCP/IP link, the connection request time-out shall be large enough to allow the connection to be established, which could involve resolving a host name, or going through multiple gateways.

Because of the large variation in connection request processing over TCP/IP, CIP routers in the connection path shall not subtract anything from the connection request timeout.

3-4.7 Connection path

The link address portion of a TCP/IP connection path segment shall be encoded within a port segment as a string of ASCII characters. The following forms shall all be supported:

- IP address in dot notation, for example “130.151.132.55” (see RFC 1117 for the format of IP addresses);
- IP address in dot notation, followed by a ":" separator, followed by the TCP port number to be used at the specified IP address;
- Host name, for example “plc.controlnet.org”. The host name shall be resolved via a DNS request to a name server (see RFC 1035 for information on host names and name resolution);
- Host name, followed by a ":" separator, followed by the TCP port number to be used at the specified host.

The port number shall be represented in either hex or decimal. Hex shall be indicated by a leading "0x". When a port number is specified, it shall be used rather than the standard port number used for the encapsulation protocol (0xAF12). Only port 0xAF12 is guaranteed to be available in an EtherNet/IP compliant device.

NOTE: Other TCP port numbers may be implemented; however, this specification does not provide a mechanism to determine which TCP port numbers are supported by a device. The use of other TCP port numbers is therefore discouraged. The guaranteed TCP port number, 0xAF12, has been reserved with the Internet Assigned Numbers Authority (IANA) for use by the encapsulation protocol.

Since port segments must be word-aligned, a pad byte may be required at the end of the string. The pad byte shall be 0x00, and shall not be counted in the Optional Address Size field of the port segment.

NOTE: Examples of port segments are shown in Table 3-4.1 (see the CIP Common specfication, for the definition of a port segment).

Table 3-4.1 – TCP/IP link address examples

Port Segment	IP address	Notes
[12][0D] [31][33][30][2E] [31][35][31][2E] [31][33][32][2E][31][00]	130.151.132.1	Multi-byte address for port 2, 13 byte string plus a pad byte
[13][12] [70][6C][63][2E] [63][6F][6E][74][72][6F][6C][6E][65][74][2E] [6F][72][67]	plc.controlnet.org	Multi-byte address for port 3, 18 byte string, no pad byte
[16][15] [31][33][30][2E] [31][35][31][2E] [31][33][32][2E] [35][35][3A] [30][78][33][32][31][30][00]	130.151.132.55:0x3210	Multi-byte address for port 6, 21 byte string plus a pad byte
[15][17] [70][6C][63][2E] [63][6F][6E][74][72][6F][6C][6E][65][74][2E] [6F][72][67][3A] [39][38][37][36][00]	plc.controlnet.org:9876	Multi-byte address port 5, 32 byte string plus a pad byte

3-4.7.1 Network connection ID

3-4.7.1.1 General

For Ethernet-TCP/IP connections, the Network Connection ID shall be a 32-bit identifier meaningful to the device that chooses it. The Network Connection ID need not be subdivided into any specific fields.

The Network Connection ID shall not be reused until the connection has been closed or has timed out. When a device restarts, it shall not reuse Network Connection IDs from previously opened connections until those connections have been closed or have timed out. A specific connection ID shall not be reused so long as there is the possibility that packets with that connection ID are present in the network.

The following two sections describe possible methods to implement unique Network Connection IDs.

3-4.7.1.2 Using an incarnation ID (informative)

This section describes one solution that prevents Connection ID reuse when a device restarts. With this solution, the Ethernet device generates Connection IDs for class 0 and class 1 connections with format shown in Figure 3-4.1.

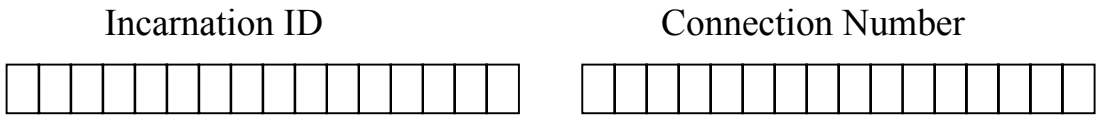


Figure 3-4.1 –Connection ID with incarnation ID

Where:

Connection Number is a 16-bit identifier meaningful to the device choosing the Connection ID.

Incarnation ID is a 16-bit identifier that each device generates before accepting or initiating any connections.

The Incarnation ID persists while the device is powered up and accepting connections. Each successive power-up cycle must cause a new (unique) Incarnation ID to be generated. The following are acceptable methods for generating Incarnation ID's:

Devices may generate a unique Incarnation ID by saving the Incarnation ID in non-volatile storage: when the device powers up, it reads the Incarnation ID from non-volatile storage. This is the Incarnation ID to use for the current cycle. It then increments the Incarnation ID and stores it for the next cycle. Note, however, that non-volatile memory devices generally have a limit to the number of times the device may be written. Depending on the device, it may not be feasible to write the Incarnation ID each powerup.

Devices may generate a unique Incarnation ID by generating a pseudo-random number at powerup. This approach requires care. By definition, there is a non-zero probability that the generated Incarnation ID is the same as the previous one. However, if done wisely, the probability is small enough to not be a concern.

Devices such as workstations, because of the large variation in startup time, can safely use the value of the system clock as an Incarnation ID. However, for embedded devices, using the system clock is not reliable since the firmware generally goes through the exact same sequence of instructions at each powerup. This results in the same clock value at the point where it would be selected for the Incarnation ID. For these embedded devices, the Incarnation ID needs to be generated based on random inputs. This is best done using a pseudo-random number generator such as the MD5 algorithm.

3-4.7.1.3 Pseudo-random connection ID per connection (informative)

This section describes another solution that prevents Connection ID reuse when a device restarts. With this solution, the device generates a pseudo-random Connection ID each time a class 0 or class 1 Connection ID is needed. The Connection ID format for this approach is shown in Figure 3-4.2.

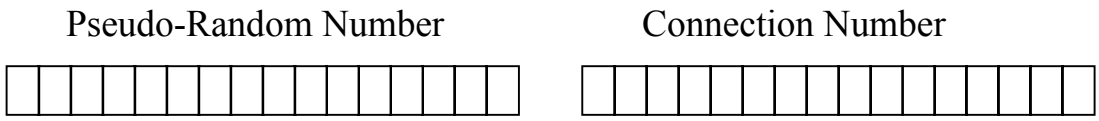


Figure 3-4.2 –Pseudo-random Connection ID

Where:

Connection Number is a 16-bit identifier meaningful to the device choosing the Connection ID.

Pseudo-Random Number is a 16-bit number generated using an appropriate pseudo-random number generator.

With this approach, the device generates the Pseudo-Random Number portion each time a Connection ID is needed. A "strong mixing function" such as the MD5 algorithm [RFC 1321] [RFC 1750] should be used to generate the pseudo-random number. Such functions take multiple input and produce pseudo-random outputs.

In order to prevent Connection IDs from being reused across powerups, the seed values for the inputs to the MD5 algorithm must be unique across successive powerup cycles. The recommended approach is to use the following inputs upon receiving the first incoming connection request:

- Vendor ID, Serial Number, Connection Serial Number
- Contents of the sockaddr_in struct (for the next hop if outbound connection; of the sender if inbound connection)
- Value of system clock

NOTE: This assumes the Ethernet device is a bridge or the Target of the connection and would not be applicable if the device is the connection Originator. For connection originators, the above seed values would likely be the same across successive powerups. Connection originators must use another source for initial seed values, or else use the Incarnation ID approach.

By definition, there is a non-zero probability that a Connection ID conflict may still occur. However, the probability is lowered by:

Using a robust pseudo-random number generator such as the MD5 algorithm.

Ensuring the seed values are different on successive power-up cycles.

3-4.8 Forward_open for CIP transport class 2 and class 3 connections

The Forward_open service for CIP class 2 and class 3 connections shall be sent over a TCP connection using the SendRRData command defined in chapter 2.

3-4.9 Forward_open for CIP transport class 0 and class 1 connections

3-4.9.1 General

The Forward_open service for CIP transport class 0 and class 1 connections shall be sent over a TCP connection using the SendRRData command defined in chapter 2. As part of the Forward_open dialog, the producer and consumer shall exchange the UDP port numbers and IP multicast address (for multicast connections) necessary to send the CIP transport class 0 and class 1 connected data. The Sockaddr Info item defined in Chapter 2 shall be used to encode the UDP port numbers and IP multicast address. The use of the Sockaddr Info item varies depending on whether the connection is multicast or point-to-point, and whether the connection originator or the connection target is the producer.

For multicast connections, the producer shall choose an IP multicast address to which to send the connected data. The port number shall be the registered UDP port number (0x08AE) assigned by the IANA. The IP multicast address and UDP port number shall be encoded via a Sockaddr Info item. A Sockaddr Info item shall be sent with Forward_open (if the connection originator is the producer), or with the Forward_open_reply (if the connection target is the producer).

For point-to-point connections, the consumer shall choose a UDP port number to which the connected data shall be sent. The port number may be the registered port number (0x08AE) or may be a port number chosen by the consumer. The port number shall be encoded in a Sockaddr Info item and shall be sent with the Forward_open (if the connection originator is the consumer), or with the Forward_open_reply (if the connection target is the consumer).

The Sockaddr Info item(s) shall be placed after the Forward_open and/or Forward_open_reply data in the SendRRData command/reply. If the Sockaddr Info items are not present, or are in error, a Forward_open_reply shall be returned with status code 0x01 and extended status 0x205.

3-4.9.2 Mapping connections to IP multicast addresses

NOTE: It is recommended, though not required, that producers use a unique IP multicast address for each active multicast connection. Depending upon the implementation, this can reduce the amount of connection screening on the part of the consumer. It also allows the consumer to more evenly service incoming connected data from multiple connections.

Since a unique IP multicast address per multicast connection is not required, consumers shall be able to handle the situation in which packets from multiple multicast connections are being sent to the same IP multicast address. Consumers shall be able to screen the incoming packets based on the Connection ID and source IP address.

NOTE: Requirements for screening connected data are defined in section 3-5.3.

3-4.9.3 Completing the multicast connection (informative)

After receiving the Forward_open_reply the consuming Ethernet devices should join the desired IP Multicast Group in order to receive the IP Multicast datagrams. The exact method for doing this depends on the TCP/IP application programming interface in use on the device.

3-5 CIP transport class 0 and class 1 connected data

3-5.1 UDP datagrams

CIP transport class 0 and class 1 packets shall be sent in UDP datagrams using the Common Packet Format defined in chapter 2. The data portion of the UDP datagram for CIP transport class 0 and class 1 packets shall be as shown in Table 3-5.1 – UDP data format for class 0 and class 1.

Table 3-5.1 – UDP data format for class 0 and class 1

Field Name	Type	Value
Item Count	UINT	2
Type ID	UINT	0x8002 (Sequenced Address Type)
Length	UINT	8
Address Data	UDINT	Connection ID (from Forward_open reply)
	UDINT	Sequence Number
Type ID	UINT	0x00B1 (Connected Data Type)
Length	UINT	Number of bytes in packet to follow
Data		class 0 or class 1 packet

3-5.2 CIP transport class 0 and class 1 packet ordering

NOTE: By definition, CIP class 0 and class 1 transports do not detect out-of-order packets. For class 0, every packet is considered to be new data. For class 1, only duplicate data is detected. If a new packet arrives and the sequence number in the transport packet is different from the previous packet, then the new packet is considered to be new data even if the new packet has a sequence number that is less than the previous sequence number.

NOTE: When using UDP to transport CIP class 0 and class 1 connected data, there is no guarantee that packets arrive in the same order that they were sent. When both sender and receiver are on the same subnet, packets typically arrive in order. However, when going through routers, when there are multiple paths that a packet could take, it is possible for packets to arrive out of order.

For class 0 and class 1 connections over Ethernet, devices shall maintain a sequence number in the UDP payload defined in section 3-5.1. The sequence number shall be maintained per connection. Each time an Ethernet device sends a CIP class 1 packet, it shall increment the sequence number for that connection. If the receiving Ethernet device receives a packet whose sequence number is less than the previously received packet, the packet with the smaller sequence number shall be discarded. Duplicate packets shall be accepted and given to the transport layer.

The sequence number shall be operated on with modular arithmetic to deal with sequence rollover.

NOTE: dealing with 32-bit sequence numbers is described in RFC793 (the TCP definition), as follows:

It is essential to remember that the actual sequence number space is finite, though very large. This space ranges from 0 to 2**32 - 1. Since the space is finite, all arithmetic dealing with sequence numbers must be performed modulo 2**32. This unsigned arithmetic preserves the relationship of sequence numbers as they cycle from 2**32 - 1 to 0 again. There are some subtleties to computer modulo arithmetic, so great care should be taken in programming the comparison of such values. The symbol "=<" means "less than or equal" (modulo 2**32).

```
Example macros show how this may be done:
/*
 * TCP sequence numbers are unsigned 32 bit integers operated
 * on with modular arithmetic.  These macros can be
 * used to compare such integers.
 */

#define SEQ_LT(a,b)     ((int)((a)-(b)) < 0)
#define SEQ_LEQ(a,b)    ((int)((a)-(b)) <= 0)
#define SEQ_GT(a,b)     ((int)((a)-(b)) > 0)
#define SEQ_GEQ(a,b)    ((int)((a)-(b)) >= 0)
```

3-5.3 Screening incoming connected data

Ethernet devices that receive class 0 and class 1 connected data shall screen incoming packets based on the network connection ID and IP address of the sending device. This is necessary for the following reasons:

- For multicast connections, there is no guaranteed mechanism to prevent multiple devices from using the same IP multicast address. Consequently, a device could receive (bogus) multicast connected data from a device with which it has not established a connection.
- For multicast connections, a device is allowed to use the same IP multicast address for multiple class 0 and class 1 multicast connections.
- To prevent network connection ID conflicts.

When a class 0 or class 1 connection is established, the target and originating Ethernet devices shall record the network connection ID on which they will receive connected data, coupled with the IP address of the device at the other end of the connection. When a device receives connected data, it shall confirm that the network connection ID is valid for the IP address of the sending device. If not, the packet shall be discarded.

3-6 IP multicast scoping and address allocation

3-6.1 Background (informative)

3-6.1.1 General

This section discusses two issues related to IP multicast that must be considered when implementing multicast connections on Ethernet: IP multicast scoping, and IP multicast address allocation.

IP multicast "scoping" refers to the practice of limiting how widely a given multicast datagram is propagated across the network. IP multicast address allocation refers to the problem of how applications obtain IP multicast addresses that they may then use to send and receive multicast datagrams.

3-6.1.2 Current scoping practices

Until Internet standards for "administrative scoping" of IP multicast traffic (see section 3-6.2) are deployed, implementations will continue to use "TTL scoping" to confine multicast traffic to some desired network boundary.

TTL scoping refers to the practice of using the TTL field in the IP header to confine multicast traffic. When sending IP multicast traffic, a host can set the TTL field in the IP header to an appropriate value based on how widely the packet should be propagated across the network. As the packet is routed across the network, the TTL field is decremented at each hop. Routers can be configured with TTL thresholds such that they will not forward a packet unless the remaining TTL is greater than the threshold.

A multicast datagram with an initial TTL of one (1) is effectively limited to a single subnet.

3-6.1.3 Current address allocation practices

The IP multicast address space is administered by the Internet Assigned Numbers Authority (IANA). The IANA does not assign addresses for the type of application usage needed by EtherNet/IP. Rather, the IANA has set aside a range of addresses from the multicast address space, and the Internet Engineering Task Force (IETF) has begun to design a set of protocols (see section 3-6.1.4) to allow administration of that address space on a per site basis.

Since the standards are evolving and are not yet deployed, an application's only real option at present is to choose its IP multicast addresses from the address space set aside by the IANA and be tolerant of the potential for clashes with other applications.

3-6.1.4 Evolving Internet standards

At present, Internet standards regarding IP multicast address allocation and scoping are evolving. However it does seem clear that the general approach that will ultimately become standard is specified in the RFC and Internet Drafts listed below (these can be found at www.ietf.org). Note that the Internet Drafts should be considered "works in progress."

D. Meyer, "Administratively Scoped IP Multicast", RFC 2365, July, 1998

The above RFC defines the administratively scoped IPv4 multicast space and describes a simple set of semantics for its administration.

Handley, Thaler, Estrin, "Malloc Architecture", draft-handley-malloc-arch-00.txt, December, 1997

The above draft proposes an architecture for multicast address allocation.

Estrin, Govindan, Handley, Kumar, Radoslavov, Thaler, "The Multicast Address-Set Claim (MASC) Protocol", draft-ietf-malloc-masc-01.txt, August, 1998.

The above draft describes a protocol that can be used for inter-domain multicast address set allocation.

Handley, "Multicast Address Allocation Protocol", draft-handley-aap-00.txt, December, 1997.

The above draft defines a protocol that addresses the specific issue of intra-domain multicast address allocation between multicast address allocation servers.

Patel, Shah, Hannah, "Multicast Address Allocation Based on the Dynamic Host Configuration Protocol", draft-ietf-malloc-mdhcp-00.txt", August, 1998.

The above draft defines the MDHCP protocol that allows hosts to request multicast address from multicast address allocation servers. MDHCP is similar to DHCP.

With the architecture specified in the above Internet drafts, Ethernet I/O devices would use MDHCP at run-time to obtain IP multicast addresses to use for I/O connections.

Once the IP multicast allocation architecture becomes standard and is deployed, this specification will be updated as appropriate.

3-6.2 Interim scoping strategy

Until the administratively scoped address architecture becomes standard and deployed, devices that send class 0 or class 1 multicast data shall use TTL scoping, and shall limit the connected datagrams to a single subnet. This is the safest approach as it prevents unwanted multicast traffic from affecting other subnets.

With this approach, if a device wishes to connect to a device on a different subnet, then a point-to-point connection shall be used.

3-6.3 Interim allocation strategy

IP multicast addresses used for multicast connections shall be in the IPv4 Organization Local Scope: 239.192.0.0/14 (note that the high-order /24 is reserved for relative assignments).

Each device capable of producing multicast data shall choose a set of IP multicast addresses to use for multicast connections. One of the following two mechanisms shall be used for choosing IP multicast addresses:

- The user may explicitly configure the device with a set of IP multicast addresses to use. The actual configuration mechanism is product-specific
- Devices may automatically choose a set of multicast addresses to use, based on their IP address, according to the following algorithm:

Each host is allowed to use a block of 32 multicast addresses, based on the host's IP address. If a subnet mask is in use, the subnet mask is applied to the IP address to obtain the relative host number: host 1 uses 239.192.1.0 through 239.192.1.31; host 2 uses 239.192.1.32 through 239.192.1.63, and so on. (Remember that the first 256 addresses in the scoped region are reserved for relative assignments, as noted above.)

To ensure that the IP multicast addresses are within the Organization Local Scope, if the subnet mask results in more than 10 bits of host address, only the low 10 bits are used. If no subnet mask is in use, then the HOST ID portion of the IP address is used. For TCP/IP class A & B IP addresses only the low-order 10 bits of the HOST ID portion of the IP address are used, making it possible that different devices could produce data on the same multicast addresses. Receiving devices must be tolerant of this (see below).

There is no mechanism to prevent multiple devices from using the same IP multicast addresses. Consequently, devices that receive connected multicast data shall screen the connected data based on the Connection ID and the IP address of the sending device, as described in section 3-5.3.

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 4: Object Model

Contents

4-1 Introduction

This chapter of the EtherNet/IP specification contains additions to the CIP object model that are EtherNet/IP specific. At this time, no such additions exist.

4-2 Scope

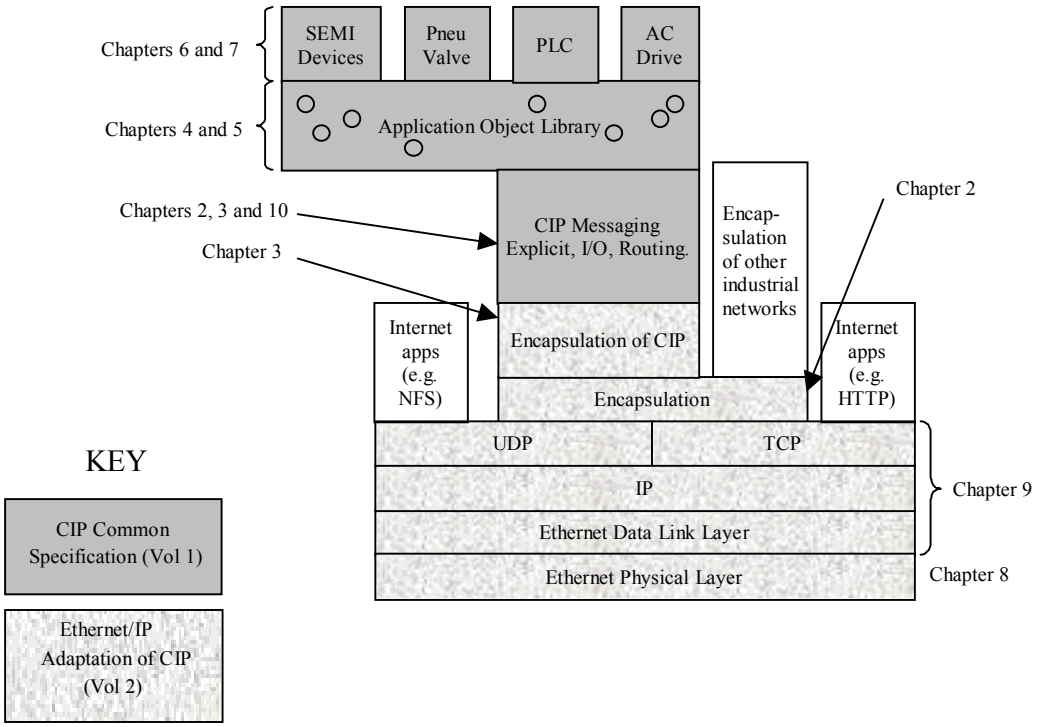


Figure 4-2.1– Document Organization Overview

This page is intentionally left blank

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 5: Object Library

Contents

5-1 Introduction

In this standard, object modeling is used to represent the network visible behavior of devices. Devices are modeled as a collection of objects. Each class of objects is a collection of related services, attributes and behaviors. Services are the procedures that an object performs. Attributes are characteristics of objects represented by values, which can vary. An object's behavior is an indication of how the object responds to particular events.

This chapter of the specification contains the object descriptions specific to EtherNet/IP. The rest of the object descriptions can be found in the CIP Common specification. With respect to the OSI reference model, CIP objects perform the Layer 7 Application functions. They also provide a mechanism to access station management counters via the network.

5-2 Reserved Class Codes

The rest of the class codes are defined in the CIP Common specification.

5-3 TCP/IP Interface Object (class code = 0xF5)

Class Code: F5hex

5-3.1 Scope

The TCP/IP Interface Object provides the mechanism to configure a device's TCP/IP network interface. Examples of configurable items include the device's IP Address, Network Mask, and Gateway Address.

The underlying physical communications interface associated with the TCP/IP Interface Object shall be any interface that supports the TCP/IP protocol. For example, a TCP/IP Interface Object may be associated any of the following: an Ethernet 802.3 interface, an ATM interface, a serial port running SLIP, a serial port running PPP, etc. The TCP/IP Interface Object provides an attribute that identifies the link-specific object for the associated physical communications interface. The link-specific object is generally expected to provide link-specific counters as well as any link-specific configuration attributes.

Each device shall support exactly one instance of the TCP/IP Interface Object for each TCP/IP-capable communications interface on the module. A request to access instance 1 of the TCP/IP Interface Object shall always refer to the instance associated with the interface over which the request was received.

5-3.2 Attributes

5-3.2.1 Class attributes

Table 5-3.1 – Class attributes

Attribute ID	Need in Implementation	Access Rule	Name	Data Type	Description of Attribute	Semantics of values
1 thru 7	These class attributes are optional and are described in Chapter 4 of Volume 1 (the CIP Common specification).					

5-3.2.2 Instance attributes

Table 5-3.2 – Instance attributes

Attr ID	Need In Implementation	Access Rule	Name	Data Type	Description of Attribute	Semantics of Values
1	Required	Get	Status	DWORD	Interface status	See section 5-3.2.2.1.
2	Required	Get	Configuration Capability	DWORD	Interface capability flags	Bit map of capability flags. See section 5-3.2.2.2.
3	Required	Set	Configuration Control	DWORD	Interface control flags	Bit map of control flags. See section 5-3.2.2.3
4	Required	Get	Physical Link Object	STRUCT of:	Path to physical link object	See section 5-3.2.2.4
			Path size	UINT	Size of Path	Number of 16 bit words in Path
			Path	Padded EPATH	Logical segments identifying the physical link object	The path is restricted to one logical class segment and one logical instance segment. The maximum size is 12 bytes. See Appendix C of Volume 1, Logical Segments.
5	Required	Set	Interface Configuration	STRUCT of:	TCP/IP network interface configuration.	See section 5-3.2.2.5
			IP Address	UDINT	The device's IP address.	Value of 0 indicates no IP address has been configured. Otherwise, the IP address shall be set to a valid Class A, B, or C address and shall not be set to the loopback address (127.0.0.1).
			Network Mask	UDINT	The device's network mask	Value of 0 indicates no network mask address has been configured.
			Gateway Address	UDINT	Default gateway address	Value of 0 indicates no IP address has been configured. Otherwise, the IP address shall be set to a valid Class A, B, or C address and shall not be set to the loopback address (127.0.0.1).

Attr ID	Need In Implementation	Access Rule	Name	Data Type	Description of Attribute	Semantics of Values
			Name Server	UDINT	Primary name server	Value of 0 indicates no name server address has been configured. Otherwise, the name server address shall be set to a valid Class A, B, or C address.
			Name Server 2	UDINT	Secondary name server	Value of 0 indicates no secondary name server address has been configured. Otherwise, the name server address shall be set to a valid Class A, B, or C address.
			Domain Name	STRING	Default domain name	ASCII characters. Maximum length is 48 characters. Shall be padded to an even number of characters (pad not included in length). A length of 0 shall indicate no Domain Name is configured.
6	Required	Set	Host Name	STRING	Host name	ASCII characters. Maximum length is 64 characters. Shall be padded to an even number of characters (pad not included in length). A length of 0 shall indicate no Host Name is configured. See section 5-3.2.2.6.

5-3.2.2.1 Status instance attribute

The **Status** attribute is a bitmap that shall indicate the status of the TCP/IP network interface. Refer to the state diagram in section 5-3.4 Behavior, for a description of object states as they relate to the Status attribute.

Table 5-3.3 – Status attribute

Bit(s):	Called:	Definition	
0-3	Interface Configuration Status	Indicates the status of the Interface Configuration attribute.	0 = The Interface Configuration attribute has not been configured. 1 = The Interface Configuration attribute contains valid configuration. 2-15 = Reserved for future use.
4-31	Reserved	Reserved for future use and shall be set to zero.	

5-3.2.2.2 Configuration Capability instance attribute

The **Configuration Capability** attribute is a bitmap that indicates the device's support for optional network configuration capability. Devices are not required to support any one particular item, however must support at least one method of obtaining an initial IP address.

Table 5-3.4 – Configuration Capability attribute

Bit(s):	Called:	Definition
0	BOOTP Client	1 (TRUE) shall indicate the device is capable of obtaining its network configuration via BOOTP.
1	DNS Client	1 (TRUE) shall indicate the device is capable of resolving host names by querying a DNS server.
2	DHCP Client	1 (TRUE) shall indicate the device is capable of obtaining its network configuration via DHCP.
3	DHCP-DNS Update	1 (TRUE) shall indicate the device is capable of sending its host name in the DHCP request as documented in Internet draft <draft-ietf-dhc-dhcp-dns-12.txt>.
4	Configuration Settable	1 (TRUE) shall indicate the Interface Configuration attribute is settable. Some devices, for example a PC or workstation, may not allow the Interface Configuration to be set via the TCP/IP Interface Object.
5-31	Reserved	Reserved for future use and shall be set to zero.

5-3.2.2.3 Configuration Control instance attribute

The **Configuration Control** attribute is a bitmap used to control network configuration options.

Table 5-3.5 – Configuration Control attribute

Bit(s):	Called:	Definition	
0-3	Startup Configuration	Determines how the device shall obtain its initial configuration at start up.	0 = The device shall use the interface configuration values previously stored (for example, in non-volatile memory or via hardware switches, etc). 1 = The device shall obtain its interface configuration values via BOOTP. 2 = The device shall obtain its interface configuration values via DHCP upon start-up. 3-15 = Reserved for future use.
4	DNS Enable	If 1 (TRUE), the device shall resolve host names by querying a DNS server.	
5-31	Reserved	Reserved for future use and shall be set to zero.	

Devices are not required to support any of the particular values of the Startup Configuration bits, however a device must support at least one method of obtaining an initial TCP/IP interface configuration.

Some devices, in particular low-end devices, may choose to obtain network interface configuration via BOOTP or DHCP only. The use of BOOTP and/or DHCP may not be appropriate for all devices. DHCP in particular supports dynamically allocated IP addresses, which could result in a device getting a different IP address each time it powers up. This behavior is not appropriate for devices that need to have static IP addresses.

A device may obtain its initial IP address via BOOTP or DHCP and then have that address retained in non-volatile storage. After receiving an IP address via BOOTP or DHCP, the address can be retained by setting the Startup Configuration bits to 0.

Out of the box, a device may wish to obtain its initial configuration via some method other than BOOTP or DHCP. For example, the device may wish to obtain its initial configuration over an attached serial port. In this case, the device should have its Startup Configuration bits set to 0, and its Interface Configuration attribute fields to all 0s. The device should then wait to be configured.

Once a device is up and running, when the value of the Startup Configuration bits is 0, a request to set the Interface Configuration attribute shall cause the device to store the contents of the Interface Configuration attribute in non-volatile storage if supported by the device. The Startup Configuration bits shall not be set to 0 unless the Interface Configuration attribute minimally contains a valid IP address. Otherwise the device could be rendered unable to communicate on the network.

Additional standard configuration methods are may be adopted in the future as they are developed and accepted by the Internet community. Non-standard techniques, including various forms of "IP Gleaning," which rely upon arrival of unusual sequences of messages shall not be used for configuration of EtherNet/IP nodes.

5-3.2.2.4 Physical Link Object

This attribute identifies the object associated with the underlying physical communications interface (e.g., an 802.3 interface). There are two components to the attribute: a Path Size (in UINTs) and a Path. The Path shall contain a Logical Segment, type Class, and a Logical Segment, type Instance that identifies the physical link object. The maximum Path Size is 6 (assuming a 32 bit logical segment for each of the class and instance).

The physical link object itself typically maintains link-specific counters as well as any link-specific configuration attributes. If the CIP port associated with the TCP/IP Interface Object has an Ethernet physical layer, this attribute shall point to an instance of the Ethernet Link Object (class code = 0xF6). For example, the path could be as follows:

Table 5-3.6 – Example path

Path	Meaning
[20][F6][24][01]	[20] = 8 bit class segment type; [F6] = Ethernet Link Object class; [24] = 8 bit instance segment type; [01] = instance 1.

5-3.2.2.5 Interface Configuration

This attribute contains the configuration parameters required to operate as a TCP/IP node. In order to prevent incomplete or incompatible configuration, the parameters making up the Interface Configuration attribute cannot be set individually. To modify the Interface Configuration attribute, the user should first Get the Interface Configuration attribute, change the desired parameters then set the attribute.

The TCP/IP Interface Object shall apply the new configuration upon completion of the Set service. If the value of the Startup Configuration bits (Configuration Control attribute) is 0, the new configuration shall be stored in non-volatile memory. The device shall not reply to the set

service until the values are safely stored to non-volatile storage. An attempt to set any of the components of the Interface Configuration attribute to invalid values (see Semantics of Values in Table 5-3.2) shall result in an error (status code 0x09) returned from the Set service.

If initial configuration is to be obtained via BOOTP or DHCP, the Interface Configuration attribute components shall be all zeros until the BOOTP or DHCP reply is received. Upon receipt of the BOOTP or DHCP reply, the Interface Configuration attribute shall show the configuration obtained via BOOTP/DHCP.

Devices are not required to support the Set service. Some implementations, for example those running on a PC or Workstation, need not support setting the network interface configuration via the TCP/IP Interface Object.

Components of the interface configuration attributes are described below:

Table 5-3.7 – Interface Configuration attribute

Name	Meaning
IP address	The device's IP address.
Network mask	The device's network mask. The network mask is used when the IP network has been partitioned into subnets. The network mask is used to determine whether an IP address is located on another subnet.
Gateway address	The IP address of the device's default gateway. When a destination IP address is on a different subnet, packets are forwarded to the default gateway for routing to the destination subnet.
Name server	The IP address of the primary name server. The name server is used to resolve host names. For example, that might be contained in a CIP connection path.
Name server 2	The IP address of the secondary name server. The secondary name server is used when the primary name server is not available, or is unable to resolve a host name.
Domain name	The default domain name. The default domain name is used when resolving host names that are not fully qualified. For example, if the default domain name is "odva.org", and the device needs to resolve a host name of "plc", then the device will attempt to resolve the host name as "plc.odva.org".

For additional information on IP addressing, subnetworks, gateways, etc. refer to Comer, Douglas E.*; Internetworking with TCP/IP, Volume 1: Protocols and Architecture*; Englewood Cliffs, NJ; Prentice-Hall, 1990.

5-3.2.2.6 Host Name

The **Host Name** attribute contains the device's host name. The host name attribute is used when the device supports the DHCP-DNS Update capability and has been configured to use DHCP upon start up. The DHCP-DNS Update mechanism is specified Internet draft <draft-ietf-dhc-dhcp-dns-12.txt>, and is supported in Windows 2000. The mechanism allows the DHCP client to transmit its host name to the DHCP server. The DHCP server then updates the DNS records on behalf of the client.

The host name attribute does not need to be set for the device to operate normally. The value of the Host Name attribute, if it is configured, shall be used for the value of the FQDN option in the DHCP request. If the Host Name attribute has not been configured then the device shall not include the FQDN option in the DHCP request.

For devices that do not support the DHCP-DNS capability, or are not configured to use DHCP, then the host name can be used for informational purposes.

5-3.3 Common services

5-3.3.1 All services

The TCP/IP Interface Object shall provide the following common services.

Table 5-3.8 – Common services

Service Code	Need in Implementation		Service name	Description of Service
	Class	Instance		
0x01	Optional	Optional	Get_Attribute_All	Returns a predefined listing of this objects attributes (See the Get_Attribute_All response definition in section 5-3.3.2)
0x02	n/a	Optional	Set_Attribute_All	Modifies all settable attributes.
0x0E	Optional	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
0x10	n/a	Required	Set_Attribute_Single	Modifies a single attribute.

5-3.3.2 Get_Attribute_All response

For class attributes, (since there is only one class attribute) class attribute #1 shall be returned.

For instance attributes, attributes shall be returned in numerical order. The Get_Attribute_All reply shall be as follows:

Table 5-3.9 – Get_Attribute_All

Attribute ID	Size in Bytes	Contents
1	4	Status
2	4	Configuration Capability
3	4	Configuration Control
4	2	Physical Link Object, Path Size
	Variable, 12 bytes max	Physical Link Object, Path (if Path Size is non-zero)
5	4	IP Address
	4	Network Mask
	4	Gateway Address
	4	Name Server
	4	Secondary Name Server
	2	Domain Name Length
	Variable, equal to Domain Name Length	Domain Name
	1	Pad byte only if Domain Name Length is odd
6	2	Host Name Length
	Variable, equal to Host Name Length	Host Name

Attribute ID	Size in Bytes	Contents
	1	Pad byte only if Host Name Length is odd

The lengths of the Physical Link Object path, Domain Name, and Host Name are not known before issuing the Get_Attribute_All service request. Implementers shall be prepared to accept a response containing the maximum sizes of the Physical Link Object path (6 UINTs), the Domain Name (48 USINTs), and Host Name (64 USINTs).

5-3.3.3 Set_Attribute_All Request

The instance Set_Attribute_All request contains the Configuration Control attribute, followed by the Interface Configuration attribute.

5-3.4 Behavior

The behavior of the TCP/IP Interface Object shall be as illustrated in the State Transition Diagram below. Note that the act of obtaining an initial executable image via BOOTP/TFTP shall not be considered within the scope of the TCP/IP Interface Object behavior. Devices are free to implement such behavior, however it shall be considered to have occurred in the "Non-existent" state.

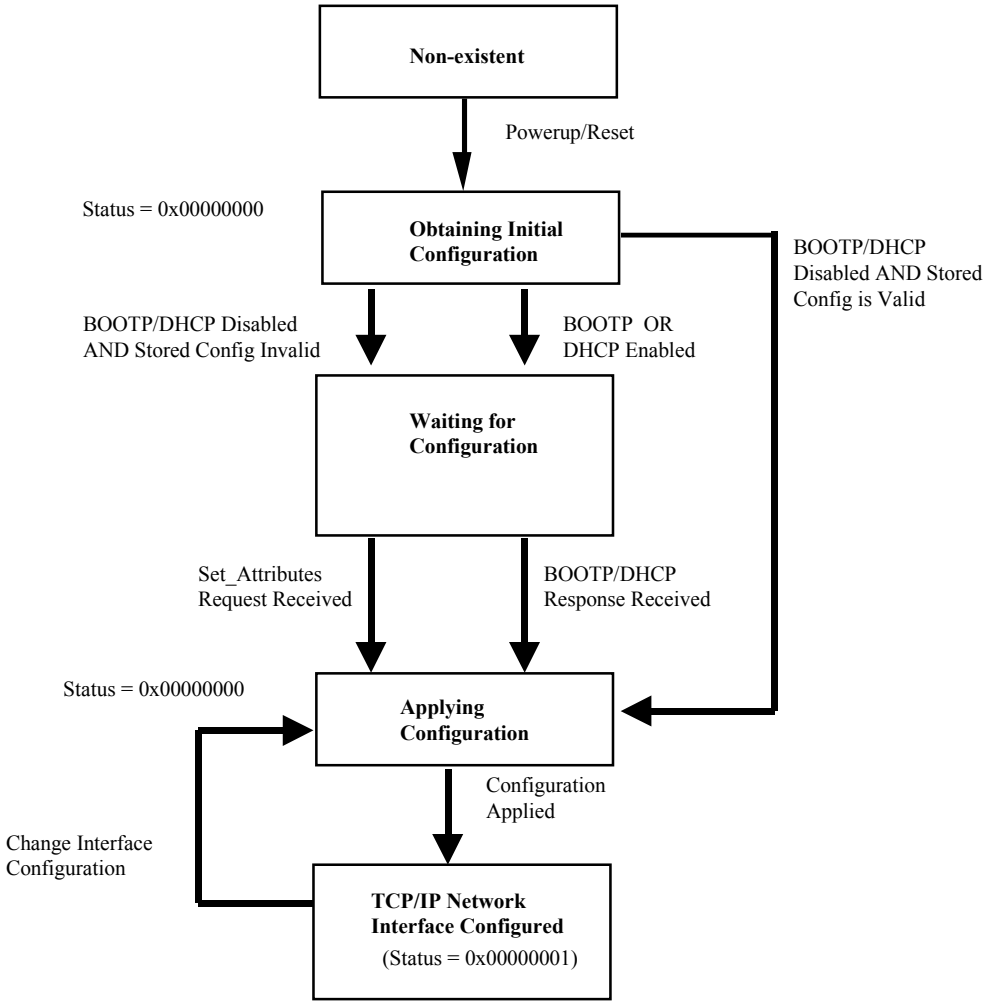


Figure 5-3.1 – State diagram showing the behavior of the TCP/IP Object

5-4 Ethernet Link Object (class code = 0xF6)

Class Code: F6hex

5-4.1 Scope

The Ethernet Link Object maintains link-specific counters and status information for a Ethernet 802.3 communications interface. Each device shall support exactly one instance of the Ethernet Link Object for each Ethernet 802.3 communications interface on the module. A request to access instance 1 of the Ethernet Link Object shall always refer to the instance associated with the communications interface over which the request was received.

5-4.2 Attributes

5-4.2.1 Class attributes

The Ethernet Link Object shall support the following class attributes.

Table 5-4.1 – Class atributes

Attribute ID	Need in Implementation	Access Rule	Name	Data Type	Description of Attribute	Semantics of values
1 thru 7	These class attributes are optional and are described in Chapter 4 of Volume 1 (the CIP Common specification).					

5-4.2.2 Instance attributes

The Ethernet Link Object shall support the following instance attributes.

Table 5-4.2 – Instance attributes

Attribute ID	Need in Implementation	Access Rule	Name	Data Type	Description of Attribute	Semantics of values
1	Required	Get	Interface Speed	UDINT	Speed of the interface	Speed in megabits per second (e.g., 10, 100, 1000, etc.)
2	Required	Get	Interface Flags	DWORD	Interface status flags	Bit map of interface flags. See section 5-4.2.2.1
3	Required	Get	Physical Address	ARRAY of 6 USINTs	MAC layer address	See section 5-4.2.2.3
4	Conditional[1]	Get	Interface Counters	STRUCT of:		See section 5-4.2.2.4
			In Octets	UDINT	Octets received on the interface	
			In Ucast Packets	UDINT	Unicast packets received on the interface	
			In NUcast Packets	UDINT	Non-unicast packets received on the interface	

Attribute ID	Need in Implementation	Access Rule	Name	Data Type	Description of Attribute	Semantics of values
			In Discards	UDINT	Inbound packets received on the interface but discarded	
			In Errors	UDINT	Inbound packets that contain errors (does not include In Discards)	
			In Unknown Protos	UDINT	Inbound packets with unknown protocol	
			Out Octets	UDINT	Octets sent on the interface	
			Out Ucast Packets	UDINT	Unicast packets sent on the interface	
			Out NUcast Packets	UDINT	Non-unicast packets sent on the interface	
			Out Discards	UDINT	Outbound packets discarded	
			Out Errors	UDINT	Outbound packets that contain errors	
5	Optional	Get	Media Counters	STRUCT of:	Media-specific counters	See section 5-4.2.2.5
			Alignment Errors	UDINT	Frames received that are not an integral number of octets in length	
			FCS Errors	UDINT	Frames received that do not pass the FCS check	
			Single Collisions	UDINT	Successfully transmitted frames which experienced exactly one collision	
			Multiple Collisions	UDINT	Successfully transmitted frames which experienced more than one collision	
			SQE Test Errors	UDINT	Number of times SQE test error message is generated	
			Deferred Transmissions	UDINT	Frames for which first transmission attempt is delayed because the medium is busy	
			Late Collisions	UDINT	Number of times a collision is detected later than 512 bit-times into the transmission of a packet	
			Excessive Collisions	UDINT	Frames for which transmission fails due to excessive collisions	

Attribute ID	Need in Implementation	Access Rule	Name	Data Type	Description of Attribute	Semantics of values
			MAC Transmit Errors	UDINT	Frames for which transmission fails due to an internal MAC sublayer transmit error	
			Carrier Sense Errors	UDINT	Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame	
			Frame Too Long	UDINT	Frames received that exceed the maximum permitted frame size	
			MAC Receive Errors	UDINT	Frames for which reception on an interface fails due to an internal MAC sublayer receive error	

[1] The Interface Counters attribute is required if the Media Counters attribute is implemented.

5-4.2.2.1 Interface Flags

The Interface Flags attribute contains status and configuration information about the physical interface and shall be as follows:

Table 5-4.3 – Interface flags

Bit(s):	Called:	Definition
0	Link Status	Indicates whether or not the Ethernet 802.3 communications interface is connected to an active network. 0 indicates an inactive link; 1 indicates an active link. The determination of link status is implementation specific. In some cases devices can tell whether the link is active via hardware/driver support. In other cases, the device may only be able to tell whether the link is active by the presence of incoming packets.
1	Half/Full Duplex	0 indicates the interface is running half duplex; 1 indicates full duplex. Note that if the Link Status flag is 0, then the value of the Half/Full Duplex flag is indeterminate.
2-31	Reserved	Shall be set to zero

5-4.2.2.2 Interface Speed

The Interface Speed attribute shall indicate whether the interface is running at 10 Mbps, 100 Mbps, 1 Gbps, etc. The scale of the attribute is in Mbps, so if the interface is running at 100 Mbps then the value of Interface Speed attribute shall be 100. The Interface Speed is intended to represent the media bandwidth; the attribute shall not be doubled if the interface is running in full-duplex mode.

5-4.2.2.3 Physical Address

The Physical Address attribute contains the interface's MAC layer address. The Physical Address is an array of octets. The recommended display format is "XX-XX-XX-XX-XX-XX", starting with the first octet. Note that the Physical Address is not a settable attribute. The Ethernet address shall be assigned by the manufacturer, and shall be unique per IEEE 802.3 requirements.

5-4.2.2.4 Interface Counters

The Interface Counters attribute contains counters relevant to the receipt of packets on the interface. These counters shall be as defined in RFC 1213 "MIB-II Management Information Base". The Interface Counters are a conditional attribute; they shall be implemented if the Media Counters attribute is implemented.

5-4.2.2.5 Media Counters

The Media Counters attribute contains counters specific to Ethernet media. These counters shall be as defined by RFC 1643, "Definitions of Managed Objects for Ethernet-Like Interface Types". If this attribute is implemented the Interface Counters shall also be implemented.

5-4.3 Common services

5-4.3.1 All services

The Ethernet Link Object shall provide the following common services.

Table 5-4.4 – Common services

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0x01	Optional	Optional	Get_Attribute_All	Returns a predefined listing of this objects attributes (See the Get_Attribute_All response definition in section 5-4.3.2)
0x0E	Conditional	Required	Get_Attribute_Single	Returns the contents of the specified attribute.

The Get_Attribute_Single shall be implemented for the class attribute if the class attribute is implemented.

5-4.3.2 Get_Attribute_All response

For class attributes, since there is only one possible attribute, the Get_Attribute_All response is the same as the Get_Attribute_Single response. If no class attributes are implemented, then no data is returned in the data portion of the reply.

For instance attributes, attributes shall be returned in numerical order, up to the last implemented attribute.

5-4.4 Class-Specific Services

The Ethernet Link Object shall support the following class-specific services:

Table 5-4.5 – Class specific services

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0x4C	n/a	Conditional[1]	Get_and_Clear	Gets then clears the specified attribute (Interface Counters or Media Counters).

[1] The Get_and_Clear service shall only be implemented if the Interface Counters and Media Counters are implemented.

5-4.4.1 Get_and_Clear Service

The Get_and_Clear service is a class-specific service. It is only supported for the Interface Counters and Media Counters attributes. The Get_and_Clear response shall be the same as the Get_Attribute_Single response for the specified attribute. After the response is built, the value of the attribute shall be set to zero.

This page is intentionally left blank

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 6: Device Profiles

Contents

6-1 Introduction

This chapter of the EtherNet/IP specification contains device profiles that are EtherNet/IP specific.

6-2 Scope

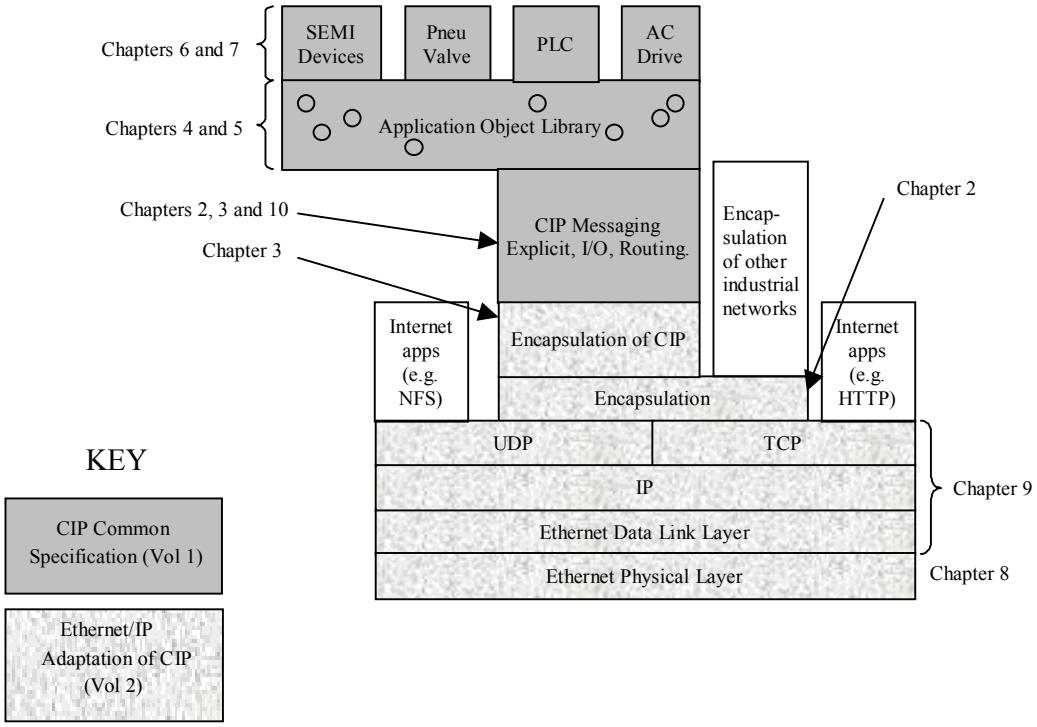


Figure 6-2.1 – Document Organization Overview

6-3 Required objects

At minimum, every EtherNet/IP device shall implement instance number one of each of the following objects:

- Identity Object (class code = 0x01)
- Message Router Object (class code = 0x02)
- Connection Manager Object (class code = 0x06)
- TCP/IP Interface Object (class code = 0xF5) and a corresponding link object

If an Ethernet medium is used, the corresponding link object shall be the Ethernet Link Object (class code = 0xF6). If any other medium is used, the vendor shall define a vendor specific link object.

NOTE: This specification permits the use of any medium that supports TCP/IP; however, only the Ethernet medium has been completely standardized here. It is likely in the future that ODVA/CI will standardize other link objects for frequently used TCP/IP media. For example, in the future, a standardized PPP object may be defined.

Although it does not have an object class code, each device shall also implement the CIP Unconnected Message Manager (UCMM)

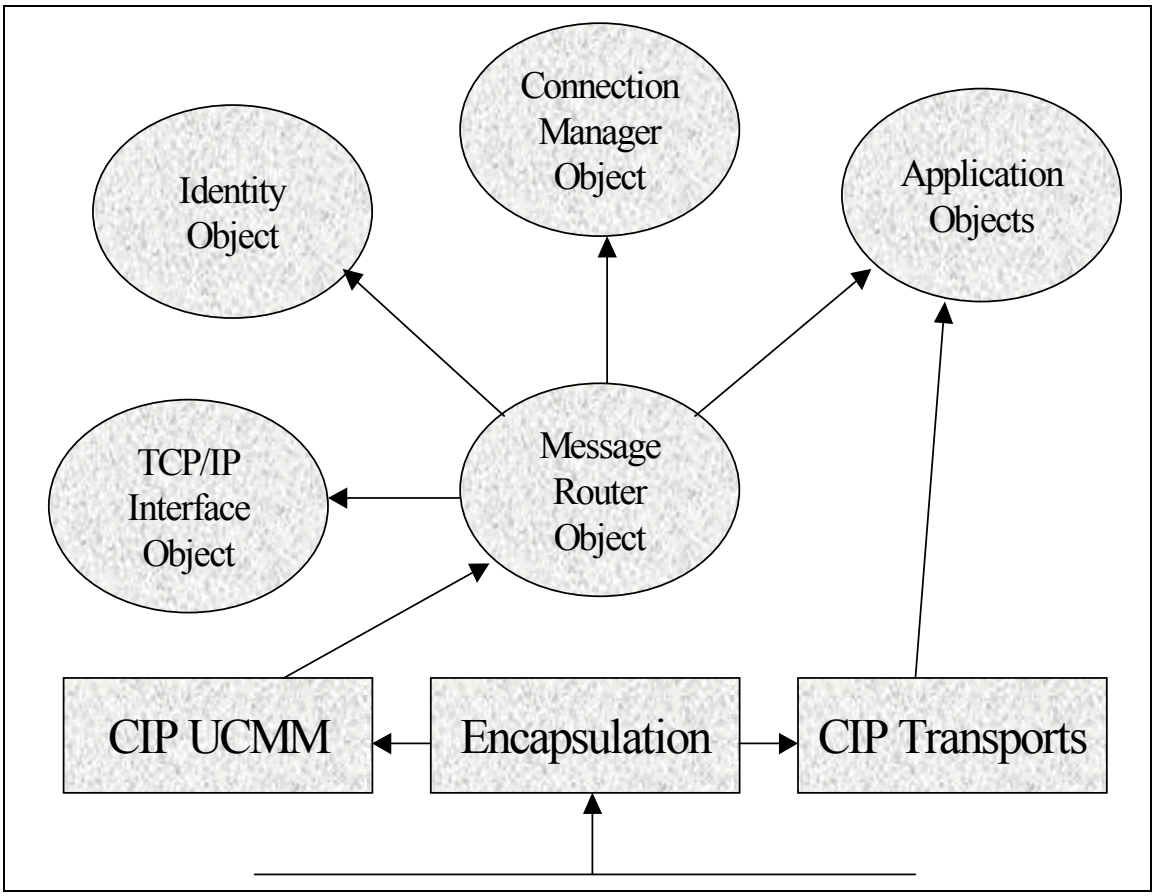


Figure 6-3.1: Base Device Object Model

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 7: Electronic Data Sheets

Contents

7-1 Introduction

This chapter of the EtherNet/IP specification contains additions to the definition of electronic data sheets (EDS) that are EtherNet/IP specific. See the CIP Common specification for more information about the format of electronic data sheets and the definition of EDS related terms such as EDS section, EDS entry and EDS field.

7-2 Scope

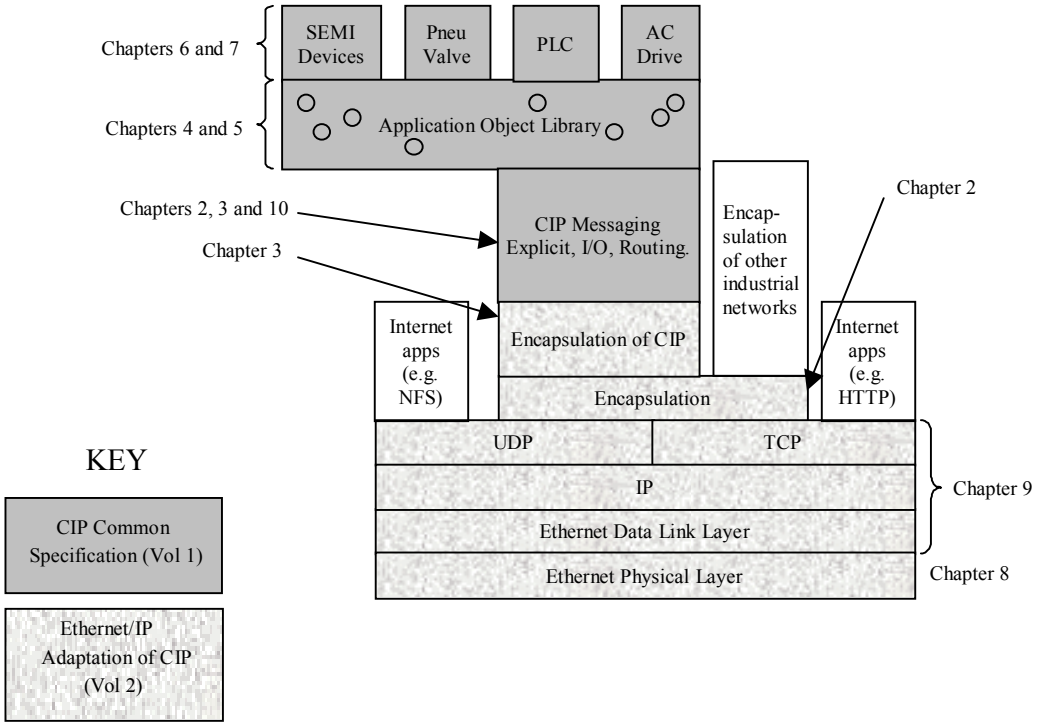


Figure 7-2.1 – Document Organization Overview

7-3 [Device Classification] section

In the [Device Classification] section of the EDS, for any EtherNet/IP compliant device, there shall be at least one ClassN keyword entry with its first field set to EthernetIP. As shown in Figure 7-4.1 – Example EDS of an EtherNet/IP device, no sub-classifications shall be present.

7-4 [Port] section

In the [Port] section of the EDS (see Figure 7-4.1 – Example EDS of an EtherNet/IP device for an example), the PortN entry corresponding to the EtherNet/IP compliant port shall be set as follows:

- The "Port Type" field shall have a value of "TCP".
- The optional "Port Object" field shall be set to the path of the TCP Object for this port.
- No additional requirements, beyond those in the CIP Common Specification (volume 1), are placed on the "Name" and "Port Number" fields. .

NOTE: An EDS for an EtherNet/IP device does not directly refer to the link object for the EtherNet/IP port (for example, the Ethernet Link Object) since it can be referenced through the TCP Object for the port.

```
[File]
      DescText = "Widget EDS File";
      CreateData = 02-07-2001;
      CreateTime = 17:51:44;
      ModDate = 04-06-1997;
      ModTime = 22:07:30;
      Revision = 2.1;
      HomeURL = "http://www.controlnet.org/EDS/12345.eds";


[Device]
      VendCode = 65535;
      VendName = "Widget-Works, Inc.";
      ProdType = 0;
      ProdTypeStr = "Generic";
      ProdCode = 10;
      MajRev = 1;
      MinRev = 1;
      ProdName = "Smart-Widget";
      Catalog = "1492-SW";
      Icon = "widget.ico";


[Device Classification]
      Class1 = EthernetIP;

[Port]
      Port1 =
            TCP,
            "EtherNet/IP port",
            "20 F5 24 01",
            1;
```

Figure 7-4.1 – Example EDS of an EtherNet/IP device

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 8: Physical Layer

Contents

8-1 Introduction

Chapter 8 specifies EtherNet/IP media and physical layer for EtherNet/IP installations. In some cases, industrial environmental requirements may exceed those in office environments. Products and components may need to be enhanced to provide the level of performance required to support industrial applications. Some of these enhancements include noise rejection, sealing, voltage isolation, chemical resistance, shock, vibration, and wide/dynamic temperature ranges.

8-2 Scope

The physical layer requirements include specifications for topology, media, connectors, shielding and grounding of components (see Figure 8-2.1). Installation requirements and practices are outlined in detail in the EtherNet/IP Planning and Installation Guide, available from ODVA/CI.

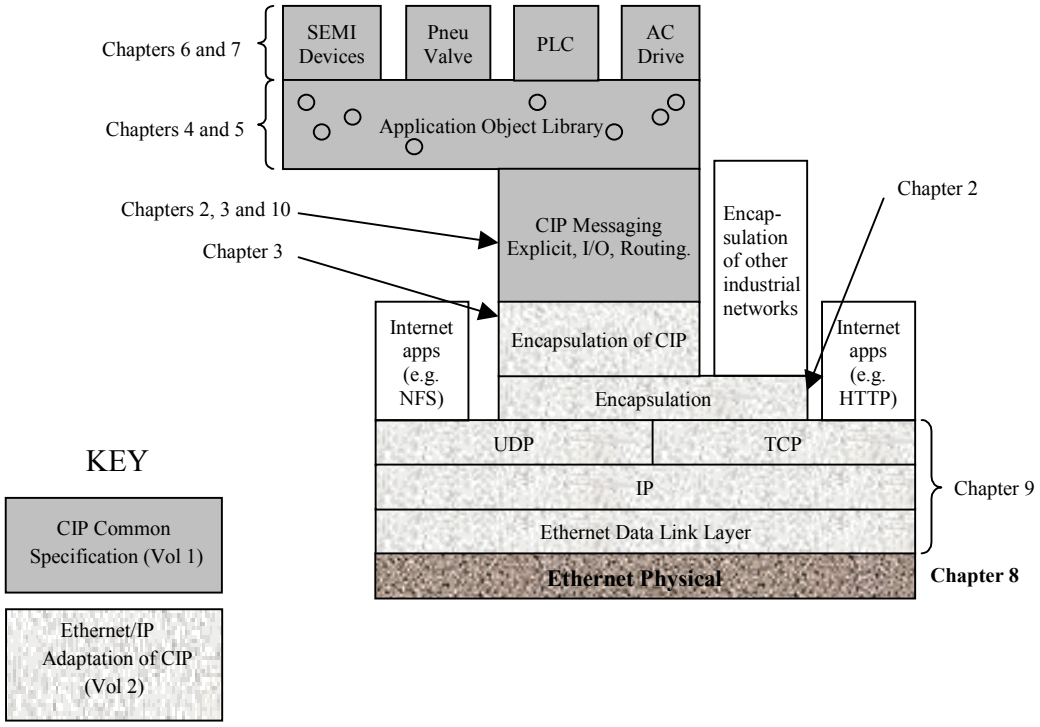


Figure 8-2.1 – Document Organization Overview

8-3 General

The following sections will delineate physical layer media variants for EtherNet/IP. This standard does not define requirements for coaxial Ethernet components or commercial off the shelf components (COTS). Coaxial and COTS systems have been deployed into the industrial environments primarily in information systems and limited control applications. These

systems, for the most part, have been successfully providing services at 10 Mb/s. Whether providing services at 10 Mb/s or 100 Mb/s, COTS components are recognized and accepted for use within the guidelines of this specification. However, testing has shown that to survive harsh environments such as high noise, diverse temperatures and the presence of chemicals, system and component enhancements are required.

This document defines component performance up to 100 Mb/s. The component specifications herein are optimized for data rates of 10 and 100 Mb/s. The copper variant shall include both shielded and unshielded twisted pair cable technologies. The signaling and coupling for copper twisted pair methods are described in section 8-6.2.

8-4 Performance Levels

Sections 8-4.1and 8-4.2 define two levels of product performance: COTS EtherNet/IP and Industrial EtherNet/IP.

NOTE: There are many sections within this chapter that specify optional requirements. This section distils these requirements into two distinct levels: commercial copper and fiber and industrial EtherNet/IP copper and fiber.

8-4.1 COTS based EtherNet/IP products

ANSI/EIA/TIA 568 A and B shall define the requirements for COTS cables. Copper and fiber-based COTS EtherNet/IP products shall meet all applicable requirements of the EtherNet/IP specification including the physical layer of Section 8-5. The use of COTS components may degrade system performance. Use of such products or components may result in unsatisfactory performance in industrial control applications.

8-4.2 Industrial EtherNet/IP products

Copper and fiber-based Industrial EtherNet/IP products shall meet all applicable requirements of the EtherNet/IP specification including section 8-6 Industrial EtherNet/IP Media and Physical Layer . For a product to achieve the Industrial EtherNet/IP performance level, the physical layer shall conform to the requirements as outlined in section 8-6 of this chapter.

8-5 COTS based EtherNet/IP Media and Physical Layer

The use of COTS components may degrade system performance. Use of such products or components may result in unsatisfactory performance in industrial control applications.

8-5.1 Copper media

8-5.1.1 Copper Media Attachment (Normative references)

A copper media attachment to an EtherNet/IP network shall include either shielded or unshielded twisted pair technology based on ANSI/TIA/EIA-568 B.2 standard. The signaling and coupling for these variants shall be as specified in the IEEE 802.3u / TP-PMD standard.

8-5.1.2 Exposing internal interfaces

The IEEE 802.3u standard defines many internal interfaces within the Physical Layer. EtherNet/IP products need not directly implement each of these interfaces, but shall behave "as if" these interfaces existed. These interfaces may be internal to the node and possibly internal to a semiconductor device.

8-5.1.3 Cables

The performance of shielded or unshielded twisted pair cables shall be based on ANSI/TIA/EIA 568-B.2 standards.

8-5.1.4 Connectors

8-5.1.4.1 COTS RJ 45 Connector Variant

The RJ45 connectors are the de-facto standard for Ethernet systems. ANSI/TIA/EIA-568 B.2 shall define the requirements for COTS RJ 45 connectors. In addition IEC 60603-7 defines the mechanical requirements for the COTS RJ 45 Connectors.

8-5.1.5 Topology Constraints

The total permanent link length for COTS twisted pair systems is limited to 90m (298 ft). The permanent link shall conform to to ANSI/TIA/EIA-568-B1.

The total channel length for COTS twisted pair systems is 100m (330 ft) including patch cables as defined in ANSI/TIA/EIA-568-B.1. Channel and patch cable design and testing shall be in accordance with ANSI/TIA/EIA-568-B.1 and B.2 respectively.

8-6 Industrial EtherNet/IP Media and Physical Layer

8-6.1 Environmental Requirements

Copper and fiber based Industrial EtherNet/IP products shall meet the environmental requirements as defined in Table 8-6.1 Minimum Environmental Specifications.

Table 8-6.1 Minimum Environmental Specifications

Environmental Test	Criteria	Industry Standard
Vibration (Unpackaged)		IEC 60068-2-6
Frequency Range	10-500Hz	
Acceleration	5g (operational)	
Displacement	0.012 inch (p-p)	
Shock (Unpackaged)		IEC 60068-2-27
Acceleration	30g (operational)	
	50g (non-operational)	
Temperature		
Operating range	-0 °C min. to +60 °C min. *	IEC 60068-2-1 IEC 60068-2-2

Environmental Test	Criteria	Industry Standard
Storage	-40 to +85 °C	IEC 60068-2-1 IEC 60068-2-2
Humidity		IEC 60068-2-30
	5 to 95% RH non-cond.	
Sealing		
	IP 20 minimum	IEC 60529
Voltage proof (connector only)		IEC 60512-1
Contact/contact	1000 Vd.c. or a.c. peak	
Contact/test panel	1500 Vd.c. or a.c. peak	

* There may be component or topology de-rating for temperatures below 0 degrees C. above 60 degrees C.

8-6.2 Copper media

8-6.2.1 Copper Media Attachment (Normative references)

A copper media attachment to an EtherNet/IP network shall support either shielded or unshielded twisted pair technology. The specifications shall contain enhancements (where needed) based on ANSI/TIA/EIA_568-B category5 cable performance levels. The signaling and coupling for these variants shall be as specified in the IEEE 802.3u / TP-PMD standard subject to the deviations listed in this section (section 8-6.2). Likewise, the cable's electrical, mechanical and environmental performance shall be as defined in section 8-6.2.2).

8-6.2.1.1 Exposing internal interfaces

The IEEE 802.3u standard defines many internal interfaces within the Physical Layer. EtherNet/IP products need not directly implement each of these interfaces, but shall behave "as if" these interfaces existed. These interfaces may be internal to the node and possibly internal to a semiconductor device.

8-6.2.2 Cable

The cable is critical in influencing the performance of the network in high noise environments. To support industrial information systems and industrial control systems there shall be two basic cable types (COTS and Industrial EtherNet/IP Cables). Only cables adhering to this specification will be eligible for the appropriate performance check mark. The two cable types will have distinct electrical/mechanical and performance requirements.

8-6.2.2.1 Industrial EtherNet/IP cables

Industrial EtherNet/IP cables shall conform to the table below.

Industrial EtherNet/IP Cable Specifications and Requirements		
Specification	**Type**	
Electrical	**Shielded**	**Unshielded**
Conductors	2 or 4 pairs + Shield	2 or 4 pairs
Attenuation	$Attenuation(f) \le 1.967\sqrt{f} + 0.023f + \frac{0.050}{\sqrt{f}}$	$Attenuation(f) \le 1.967\sqrt{f} + 0.023f + \frac{0.050}{\sqrt{f}}$
Impedance (Fitted) ASTM 4566	95 -110 Ω 1-4 Mhz 95 - 107 Ω 4 – 100 MHz	95 -110 Ω 1-4 Mhz 95 - 107 Ω 4 – 100 MHz
RL	1-10 MHz 20 + 6 Log_{10}(f) 10-20 MHz 26 20-100 MHz 26-5*Log_{10}(f/20)]	1-10 MHz 20 + 6 Log_{10}(f) 10-20 MHz 26 20-100 MHz 26-5*Log_{10}(f/20)]
NEXT Loss	NEXT(f) ≥ 64 – 15*log_{10}(f) dB	NEXT(f) ≥ 64 - 15*log_{10}(f) dB
Shielding Effectiveness	TBD	N/A
Cupp	<= 150pf/100meter	< = 150pf /100meter
DCR	9.38 Ω/100 meters	9.38 Ω/100 meters
Common Mode Rejection	TBD	TBD
DCR Unbalance	5%	5%

Mechanical	**Shielded**	**Unshielded**
Minimum Pulling Tension	25 lbs	25 lbs
Breaking Strength	400 N Min	400 N Min
Minimum Bend Radius	1" at -20C	1" at –20C

Dimensional (Recommended for RJ 45 compatibility)	**Shielded**	**Unshielded**
Jacket OD (sheath over twisted pairs)	0.250" Max	0.250" Max
Insulated Conductor	0.048" Max	0.048" Max

Additionally, the cable should provide attenuation above 62.5 MHz as described in the following paragraph. The graph can be reproduced using the standard cable attenuation formula described in TIA/EIA 568A document and repeated here, by substituting the K factors for 0, 0.3 and 0.1.

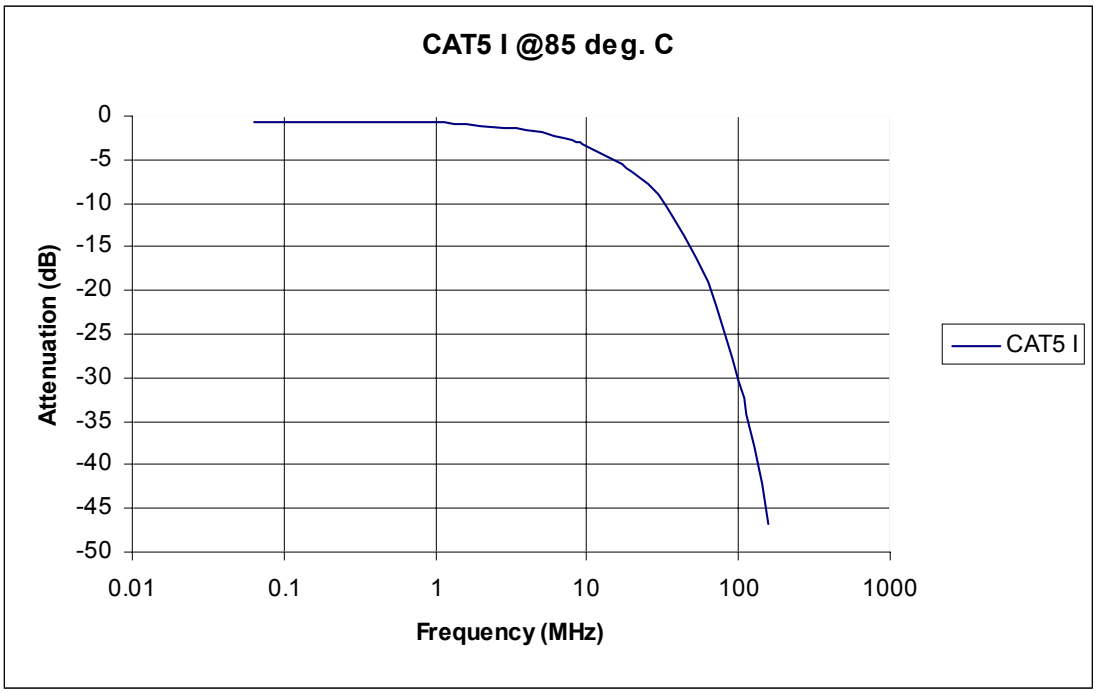
Limiting the cable bandwidth can increase the link performance in high noise environments. Increasing the cable loss above 62.5 MHz is an efficient method for limiting the noise bandwidth. The formula and attenuation factors for both standard and the industrial hardened cables are listed below (Equation 8-6.1 UTP/Shielded Attenuation.)

Equation 8-6.1 UTP/Shielded Attenuation

$$\text{Atten}_{\text{UTPCable}}(f) = \frac{-\left(K_0 \cdot \sqrt{f} + K_1 \cdot f + \frac{K_2}{\sqrt{f}}\right)}{100}$$

$K_{0-2} = (1.967 \;\; 0.023 \;\; 0.050)$ Standard, Cat 5E spec

$K_{0-2} = (0 \;\; 0.3 \;\; 0.1)$ Industrial Cable



8-6.2.3 Connectors

8-6.2.3.1 Industrial EtherNet/IP Connector RJ 45 Variant

Attachment to the medium shall be via either of two types of Industrial grade RJ-45 connectors:

- Non-Sealed industrial RJ 45 EtherNet/IP connector. The standard industrial EtherNet/IP connector shall be required to adhere to the specifications in section 8-6.2.3.1.1
- Sealed Industrial EtherNet/IP RJ 45 connector. The IP67 sealed industrial EtherNet/IP connector shall be required to adhere to the specifications in sections 8-6.2.3.1.1 and 8-6.2.3.1.2

8-6.2.3.1.1 Non-Sealed Industrial RJ-45 EtherNet/IP connector

Standard industrial hardened RJ-45 connector shall meet the following specifications:

Industrial Enet Connector Specifications and Requirements		
Specification	**Type**	
Electrical	**RJ-45-Shielded**	**RJ-45**
Conductors	8 + 1 Shield	8
Insertion Loss	ANSI/TIA/EIA-568-B.2 Category 5E	ANSI/TIA/EIA-568-B.2 Category 5E
RL	ANSI/TIA/EIA-568-B.2 Category 5E	ANSI/TIA/EIA-568-B.2 Category 5E
NEXT Loss	ANSI/TIA/EIA-568-B.2 Category 5E	ANSI/TIA/EIA-568-B.2 Category 5E
Shielding Effectiveness	ANSI/TIA/EIA-568-B.2 Category 5E	N/A

Mechanical	**RJ-45-Shielded**	**RJ-45**
Gender	Plug and Socket	Plug and Socket
Mating Specification	CEI IEC 60603-7	CEI IEC 60603-7
Contact plating	50u inches min. gold over 100u inches min. nickel or equivalent plating system	50u inches min. gold over 100u inches min. nickel or equivalent plating system
Contact LLCR over life	< 20 mΩ	< 20 mΩ
Initial Contact LLCR	<=2.5 mΩ	<=2.5 mΩ
Contact Life	750 insertions and extractions min.	750 insertions and extractions min.

8-6.2.3.1.2 Sealed Industrial EtherNet/IP RJ-45 connector variant

The sealing interface shall meet a minimum of IP67 sealing performance as defined in IEC 60529.

8-6.2.3.1.2.1 Sealed Industrial EtherNet/IP RJ 45 Jack

The Sealed RJ 45 variant is based on the VG 95 234 specification. The following sealed jack drawing sufficiently defines the jack to maintain compatibility for mating and sealing amongst various vendors who may make one or both parts. The jack may be offered as a PCB mount, bulkhead and cable end either field installed or as manufactured assembly. The jack is fully compatible with standard off-the-shelf plugs.

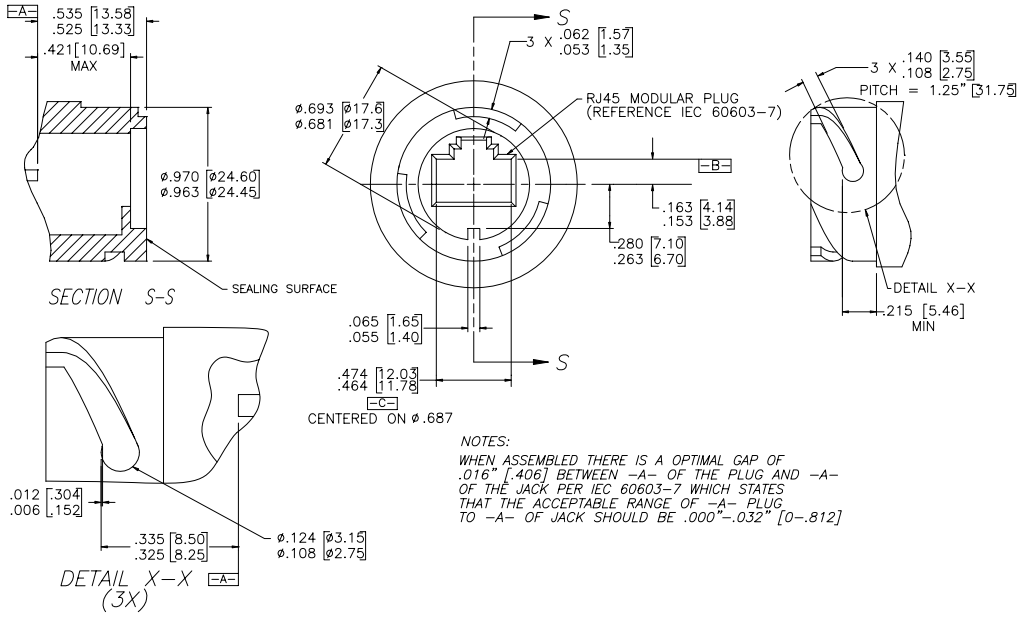


Figure 8-6.1 – Sealed jack

8-6.2.3.1.2.2 Sealed Industrial EtherNet/IP RJ 45 Plug

The Sealed RJ 45 variant is based on the VG 95 234 specification. The following sealed plug drawing sufficiently defines the plug to maintain compatibility for mating and sealing amongst various vendors who may make one or both parts. The plug may be offered as a field installable or manufactured cable assembly. The plug is compatible with standard off-the-shelf jacks with the exception of the locking mechanism.

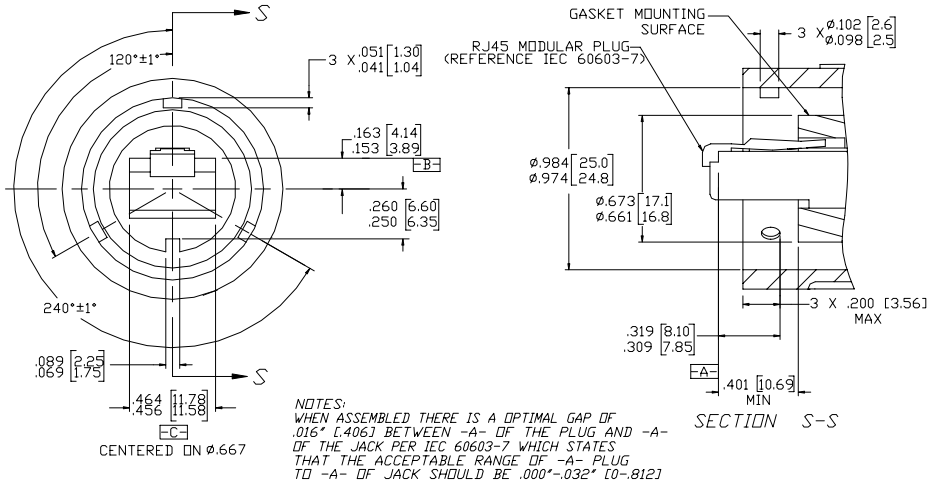


Figure 8-6.2 – RJ-45 sealed plug

8-6.2.3.2 Other connectors

Under consideration

8-6.2.3.2.1 Non sealed Other connectors

Under consideration

8-6.2.3.2.2 Sealed Other Connectors

Under consideration

8-6.2.4 Industrial EtherNet/IP Physical Layer Media Interface

A device that connects to the Industrial EtherNet/IP copper media shall conform to IEC 8802.3.

The impedance at the media interface shall conform to ISO/IEC 8802.3 (ANSI/IEEE Std 802.3) and IEEE Std 802.3u-1995 supplement with the exception of impedance tolerance. The impedance tolerance shall be limited to 5%. The temperature range and vibration shall be consistent with the targeted environment.

8-6.2.4.1 Topology constraints

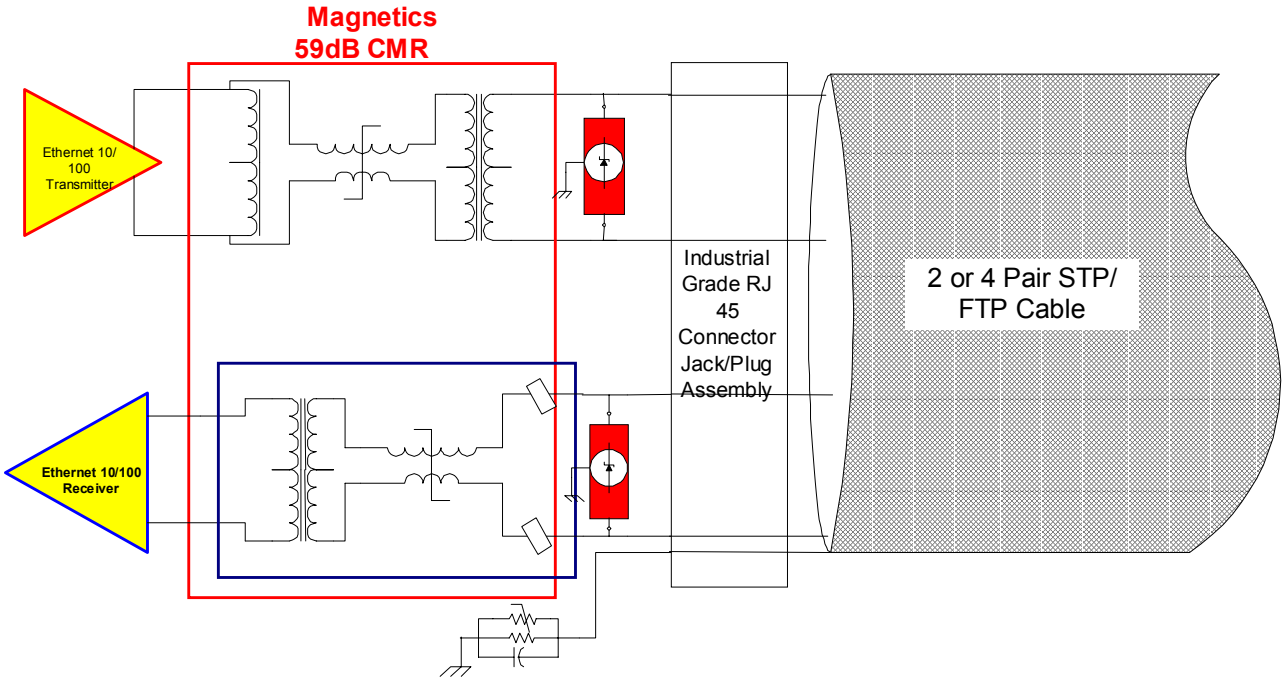
The total channel length for RJ-45 systems is 100m (330 ft) including patch cables as defined in ANSI/TIA/EIA-568-B.1. Channel and patch cable design and testing shall be in accordance with ANSI/TIA/EIA-568-B.1 and B.2 category 5E.

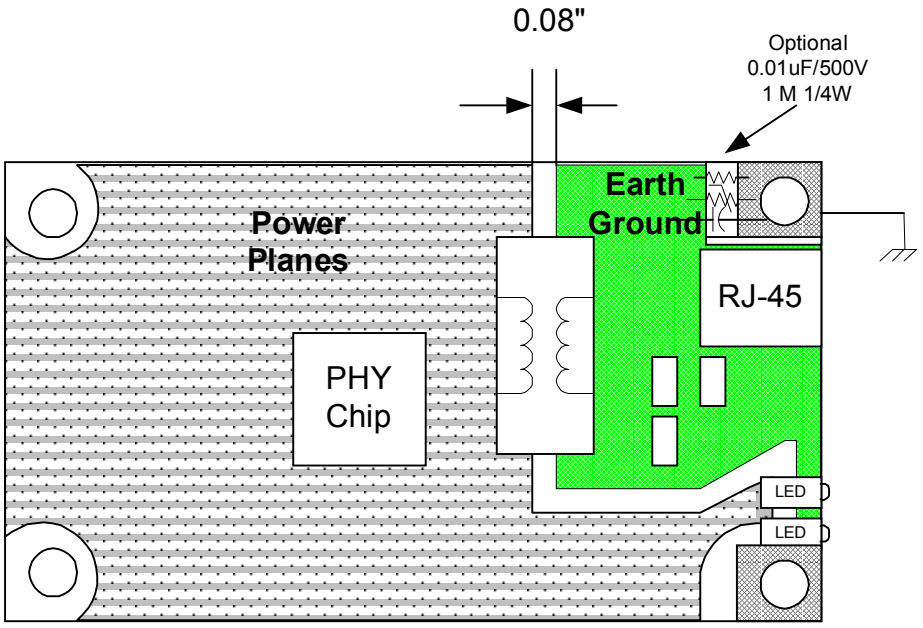
Channel design and testing for other connector systems is under consideration.

8-6.2.5 Industrial EtherNet/IP Components

In order to maximize performance in noise, it is critical that the components selected for the Media and Physical Layers provide key characteristics. The Transformer should (highly recommended) provide a minimum of 59dB common mode rejection (CMR) at 30 MHz.

The following reference block diagrams are included to help produce uniform designs.





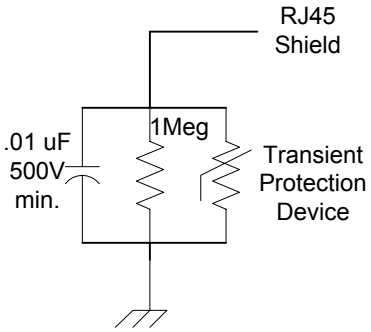
8-6.2.5.1 Shield Grounding

8-6.2.5.1.1 Connectivity device (Switch, Hub, Bridges, Routers)

All cable shields shall be terminated in either of two ways:

1) The communications shield shall be terminated directly to earth ground.

2) The communications shield shall be terminated to earth ground via a parallel resistor and capacitor combination.

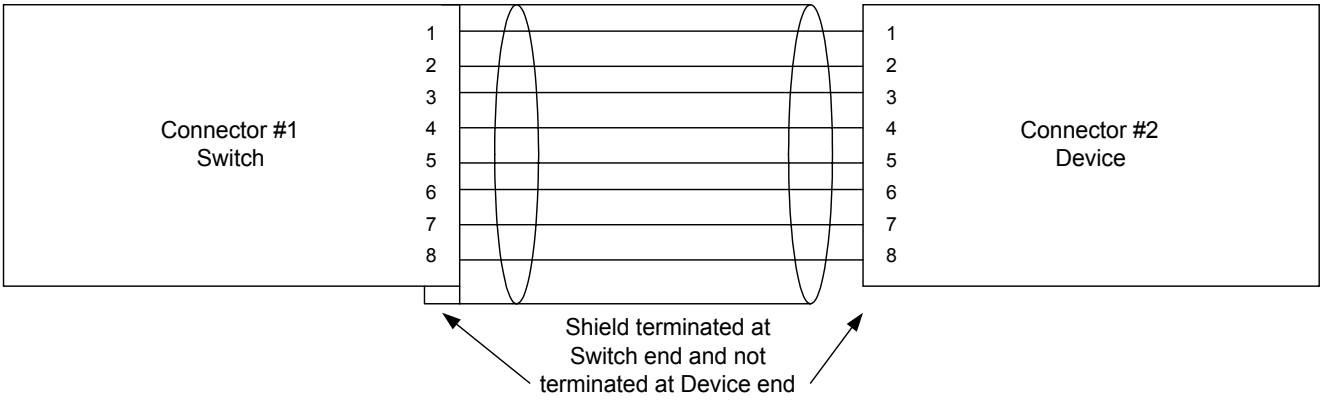


8-6.2.5.1.2 Device (sensor, PLC)

To prevent ground loops caused by shield cables, devices shall not connect the shield directly to ground. All devices shall have the shield terminated in one of two ways:

1) The shield of a device should be terminated via a parallel resistor capacitor combination. (This is the same as option 2 for the hub/switch.)

2) If the device provides direct connection to ground through the RJ45 connector, then the shield shall not be connected at the RJ45 plug.



8-6.3 Fiber media variant

NOTE: The fiber Physical Layer variant can be implemented to allow certification with the following requirements for operation in explosive atmospheres. These requirements can be met without sacrificing distance or reducing the number of nodes.

European Committee for Electromechanical Standardization (CENELEC)

European Community Requirements

A fiber media attachment to an EtherNet/IP network shall be limited to the SC, ST, MTRJ connector variants. The signaling and coupling for this variant shall be as specified in the IEEE 802.3u standard subject to the deviations listed in this section (section 8-6.3).

8-6.3.1 Connector

Fiber optic connector designs shall be either MT-RJ, SC or ST types, and comply with ANSI/TIA/EIA 568-B.3, titled Fiber Optic Cabling Components Standard.

Further, connector designs shall meet the requirements of the corresponding ANSI/TIA/EIA (Fiber Optic Connector Intermateability Standard (FOCIS) documents.

8-6.3.2 Fiber Optic Cable

Single mode and multimode optical fiber cable shall comply with the ANSI/EIA/TIA 568-B.3, titled Fiber Optic Cabling Components Standard.

8-6.3.3 Exposing internal interfaces

The IEEE 802.3u standard defines many internal interfaces within the Physical Layer. EtherNet/IP products need not directly implement each of these interfaces, but shall behave "as if" these interfaces existed. These interfaces may be internal to the node and possibly internal to a semiconductor device.

8-6.3.4 Topology constraints

8-6.3.5 Reference design (informative)

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 9: Indicators & Middle Layers

Contents

9-1 Introduction

Chapter 9 specifies the standard appearance and behavior of EtherNet/IP diagnostic LEDs. This chapter also specifies TCP/IP requirements of EtherNet/IP devices.

Scope

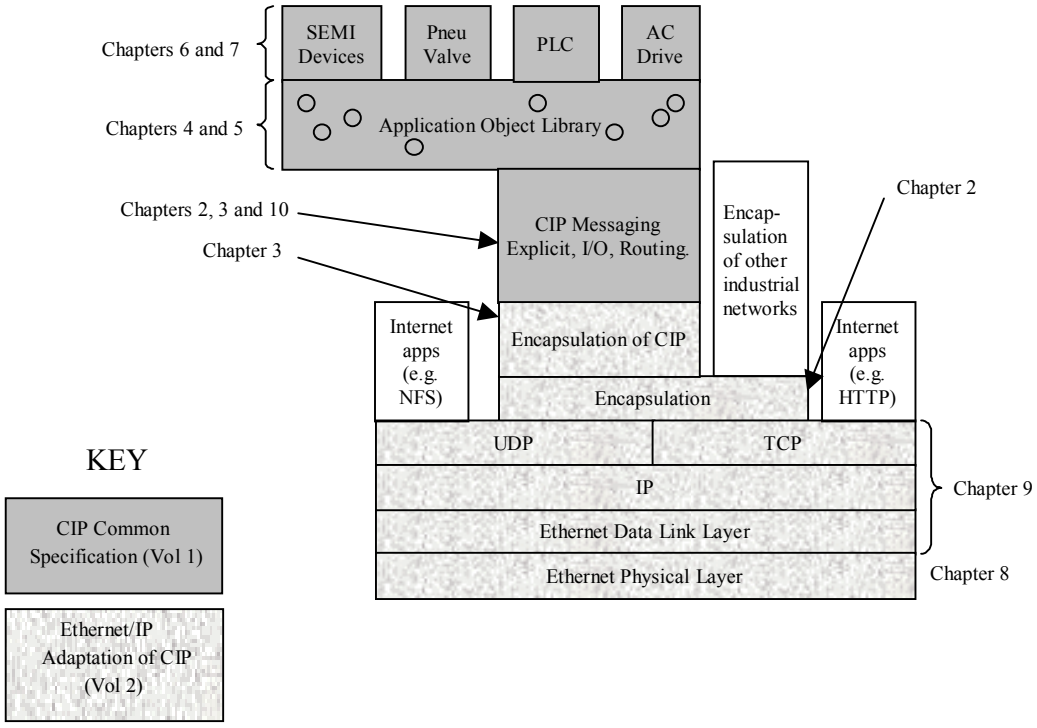


Figure 9-1.1 Document Organization Overview

9-2 Data Link Layers

Though this specification is called "EtherNet/IP", Ethernet is technically not required. The EtherNet/IP protocol may be used on any media that supports the transmission of the Internet Protocol.

NOTE: For example, the EtherNet/IP protocol could be used over FDDI, modem lines (SLIP or PPP), ATM, etc.

When any particular medium is used, it shall be used in accordance to commonly accepted standards. In particular, when Ethernet is used, it shall be used as defined by the IEEE 802.3 specification.

9-3 Requirements for TCP/IP Support

In addition to the various requirements set forth in this specification, all EtherNet/IP hosts are required to have a minimally functional TCP/IP protocol suite and transport mechanism. The minimum host requirements for EtherNet/IP hosts shall be those covered in RFC-1122, RFC-1123, and RFC-1127 and the subsequent documents that may supersede them. Whenever a feature or protocol is implemented by an EtherNet/IP host, that feature shall be implemented in accordance to the appropriate RFC documents, regardless of whether the feature or protocol is considered required or optional by this specification. The Internet and the RFCs are dynamic. There will be changes to the RFCs and to the requirements included in this section as the Internet and this specification evolves and these changes will not always provide for backward compatibility.

All EtherNet/IP devices shall at a minimum support:

- Internet Protocol (IP version 4) (RFC 791)
- User Datagram Protocol (UDP) (RFC 768)
- Transmission Control Protocol (TCP) (RFC 793)
- Address Resolution Protocol (ARP) (RFC 826)
- Internet Control Messaging Protocol (ICMP) (RFC 792)
- Internet Group Management Protocol (IGMP) (RFC 1112 & 2236)
- IEEE 802.3 (Ethernet) as defined in RFC 894

NOTE: Although the encapsulation protocol is suitable for use on other networks besides Ethernet that support TCP/IP and products may be implemented on these other networks, conformance testing of EtherNet/IP products is limited to those products on Ethernet. Other suitable networks include:

Point to Point Protocol (PPP) (RFC 1171)
ARCNET (RFC 1201)
FDDI (RFC 1103)

NOTE: EtherNet/IP devices are encouraged but not required to support other Internet protocols and applications not specified here. For example, may support HTTP, Telnet, FTP, etc. This specification makes no requirements with regards to these protocols and applications.

9-4 Indicators

9-4.1 Required indicators

A product need not have indicators to be compliant with this specification. However, to be compliant with the Industrial Conformance Level described in Chapter 8, a product shall support both the module status and network status indicators as defined by sections 9-4.2, 9-4.3 and 9-4.4.

If a product does support any of the indicators described here, they must adhere to the specifications described in this section (section 9-4).

Two types of status indicators may be provided:

- One module status indicator;

- One network status indicator;

Additional indicators may be present; however, the naming and symbol conventions of the standard indicators shall not be employed for other indicators.

NOTE: Indicators, typically implemented as LEDs, help maintenance personnel to quickly identify a faulty unit or media. As such, red indicators are used to indicate a fault condition.

NOTE: Products are encouraged to have an indicator that displays the state of link (for example, link status, tx/rx, collision, etc.) following generally accepted industry practices (as used in devices such as switches).

9-4.2 Common indicator requirements

9-4.2.1 Applicability of common requirements

The common indicator requirement shall only apply to indicators for which requirements are specified in this standard.

9-4.2.2 Visibility of indicators

Indicators shall be viewable without removing covers or parts from the equipment. Indicators shall be easily seen in normal lighting. Any labels and icons shall be visible whether or not the indicator is illuminated.

9-4.2.3 Indicator flash rate

Unless otherwise indicated, the flash rate of all indicators is approximately 1 flash per second. The indicator should be on for approximately 0.5 second and off for approximately 0.5 second. This flash rate specification only applies to the indicators specified in this chapter.

9-4.2.4 Indicators at power up

An indicator test is to be performed at power-up. To allow a visual inspection, the following sequence shall be performed:

- Turn first indicator Green, all other indicators off
- Leave first indicator on Green for approximately 0.25 second
- Turn first indicator on Red for approximately 0.25 second
- Turn first indicator on Green
- Turn second indicator (if present) on Green for approximately 0.25 second
- Turn second indicator (if present) on Red for approximately 0.25 second
- Turn second indicator (if present) Off

If other indicators are present, test each indicator in sequence as prescribed by the second indicator above. If a Module Status indicator is present, it shall be the first indicator in the sequence, followed by any Network Status indicators present. After completion of this power up test, the indicator(s) shall turn to a normal operational state.

9-4.3 Module status indicator

9-4.3.1 Description

The indication of module status shall require a single bicolor (red/green) indicator that represents the state of the entire product.

NOTE: A product with more than one communication port would have only one module status indicator, but more than one network status indicator (one per port).

9-4.3.2 Labeling

The module status indicator shall be labeled with one of the following:

- “MS”;
- "Mod";
- “Mod Status”;
- “Module Status”.

9-4.3.3 States

The module status indicator shall be in one of the following states:

Table 9-4.1 – Module status indicator

Indicator state	Summary	Requirement
Steady Off	No power	If no power is supplied to the device, the module status indicator shall be steady off.
Steady Green	Device operational	If the device is operating correctly, the module status indicator shall be steady green.
Flashing Green	Standby	If the device has not been configured, the module status indicator shall be flashing green.
Flashing Red	Minor fault	If the device has detected a recoverable minor fault, the module status indicator shall be flashing red. **NOTE**: An incorrect or inconsistent configuration would be considered a minor fault.
Steady Red	Major fault	If the device has detected a non-recoverable major fault, the module status indicator shall be steady red.
Flashing Green / Red	Self-test	While the device is performing its power up testing, the module status indicator shall be flashing green / red.

9-4.4 Network status indicator

9-4.4.1 Description

The indication of network status shall require a single bicolor (red/green) indicator that represents the state of a single communication port.

NOTE: A product with more than one communication port would have only one module status indicator, but more than one network status indicator (one per port).

9-4.4.2 Labeling

The network status indicator shall be labeled with one of the following:

- “NS”;
- “Net”;
- “Net Status”;
- "Network Status".

9-4.4.3 States

The network status indicator states shall be as follows:

Table 9-4.2 – Network status indicator

Indicator state	Summary	Requirement
Steady Off	Not powered, no IP address	If the device does not have an IP address (or is powered off), the network status indicator shall be steady off.
Flashing Green	No connections	If the device has no established connections, but has obtained an IP address, the network status indicator shall be flashing green.
Steady Green	Connected	If the device has at least one established connection (even to the Message Router), the network status indicator shall be steady green.
Flashing Red	Connection timeout	If one or more of the connections in which this device is the target has timed out, the network status indicator shall be flashing red. This shall be left only if all timed out connections are reestablished or if the device is reset.
Steady Red	Duplicate IP	If the device has detected that its IP address is already in use, the network status indicator shall be steady red.
Flashing Green / Red	Self-test	While the device is performing its power up testing, the network status indicator shall be flashing green / red.

This page intentionally left blank

Volume 2: EtherNet/IP Adaptation of CIP

Chapter 10: Bridging & Routing

Contents

10-1 Introduction

This chapter of the EtherNet/IP specification contains additions to the definition of CIP bridging and routing that are EtherNet/IP specific. At this time, no such additions exist.

10-2 Scope

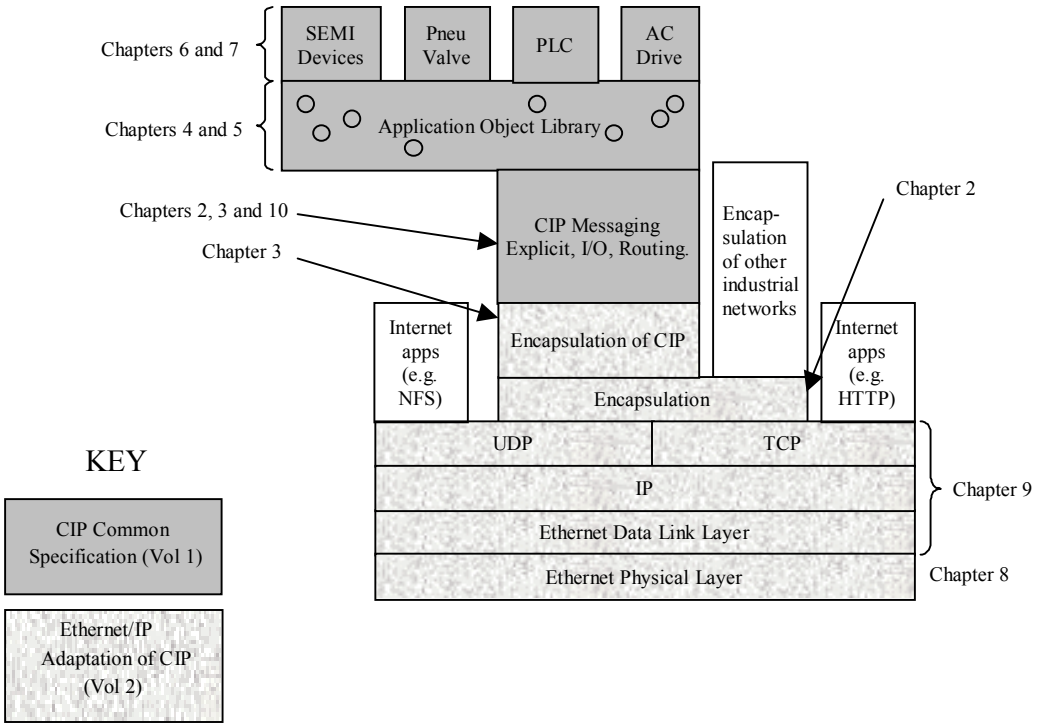


Figure 10-2.1 – Document Organization Overview

This page is intentionally left blank

Volume 2: EtherNet/IP Adaptation of CIP

Appendix A: Explicit Messaging Services

Contents

A-1 Introduction

This chapter of the EtherNet/IP specification contains additions to the definition of CIP explicit messaging services that are EtherNet/IP specific. At this time there are no such additions.

A-2 Scope

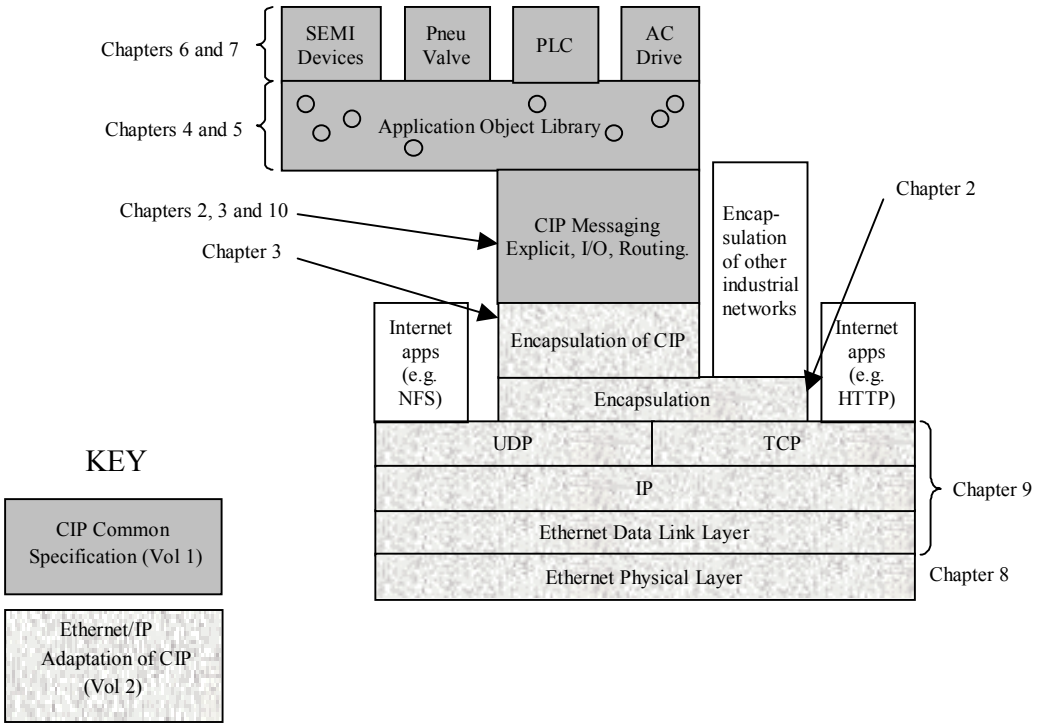


Figure A-2.1 – Document Organization Overview

This page is intentionally left blank

Volume 2: EtherNet/IP Adaptation of CIP

Appendix B: Status Codes

Contents

B-1 Introduction

This appendix of the EtherNet/IP specification contains additions to the definition of CIP error codes that are EtherNet/IP specific. At this time there are no such additions.

B-2 Scope

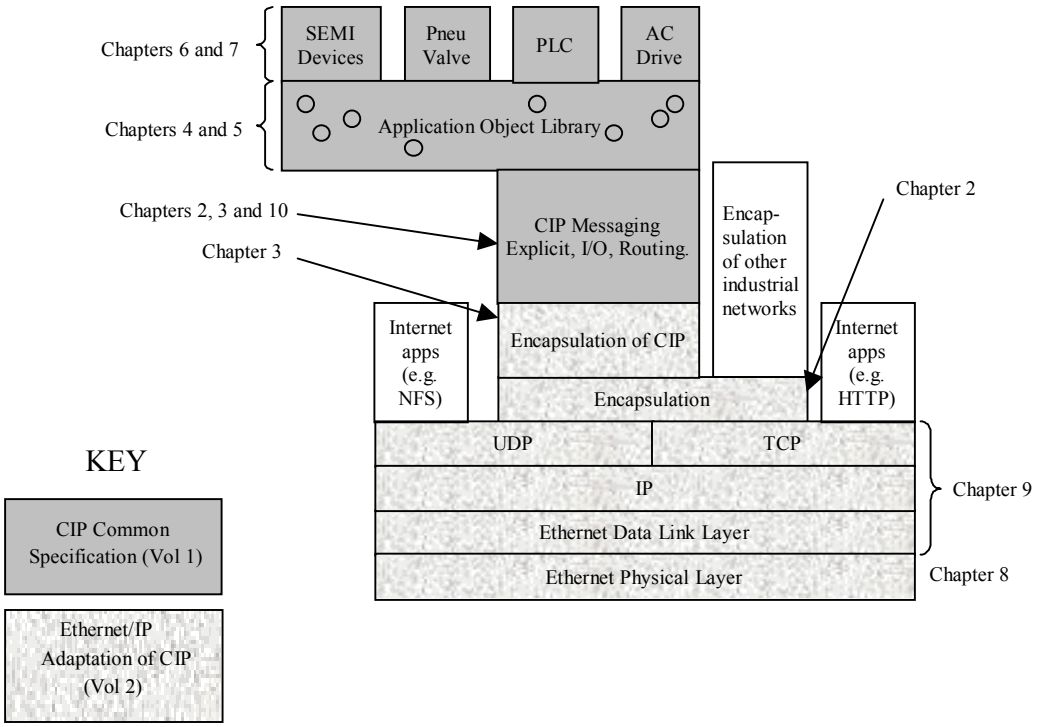


Figure B-2.1 – Document Organization Overview

This page is intentionally left blank

Volume 2: EtherNet/IP Adaptation of CIP

Appendix C: Data Management

Contents

C-1 Introduction

This chapter of the EtherNet/IP specification contains additions to the CIP Data Management specification that are EtherNet/IP specific. At this time there are no such additions.

C-2 Scope

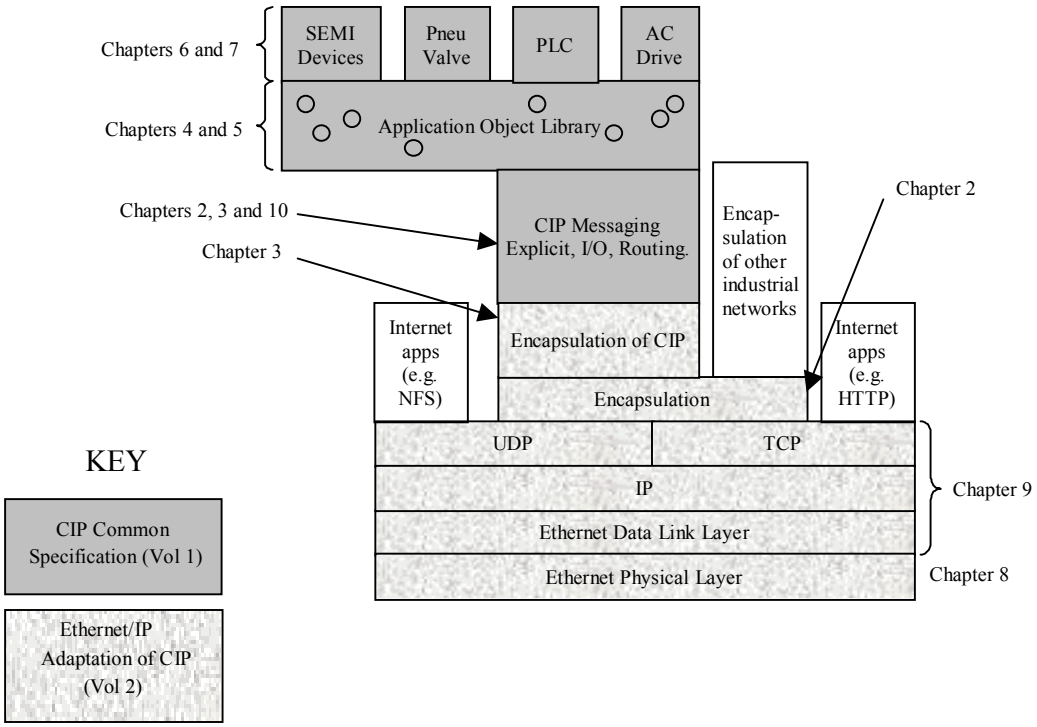


Figure C-2.1 – Document Organization Overview

This page is intentionally left blank

Volume 2: EtherNet/IP Adaptation of CIP

Appendix D: Engineering Units

Contents

D-1 Introduction

This chapter of the EtherNet/IP specification contains additions to the list of CIP engineering units that are EtherNet/IP specific. At this time there are no such additions.

D-2 Scope

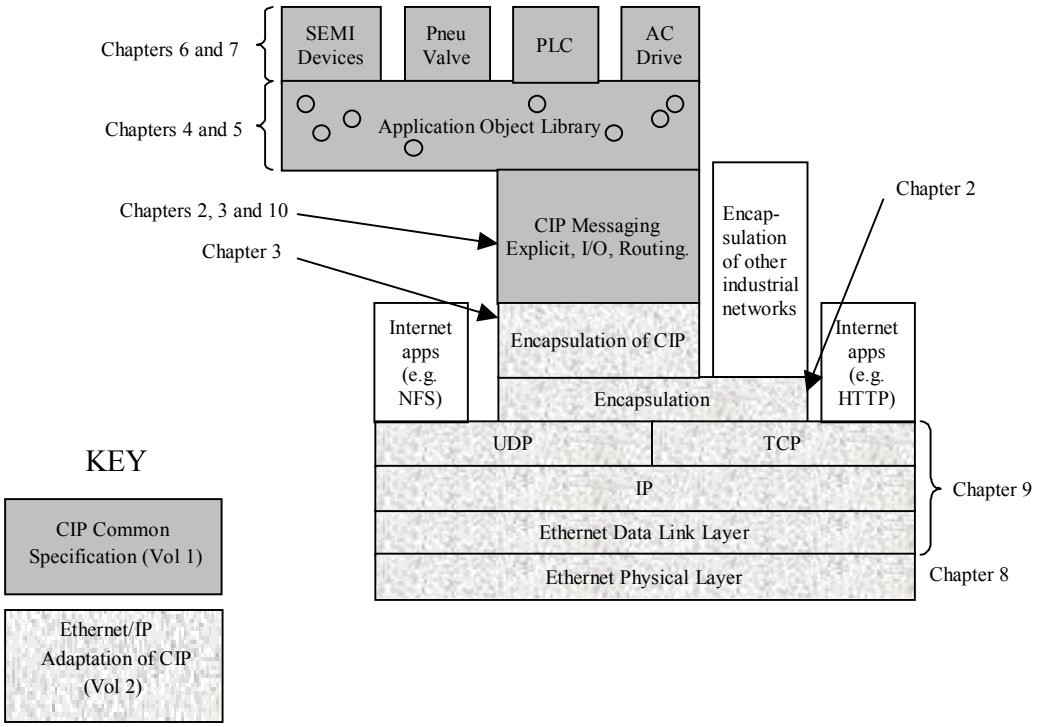


Figure D-2.1 – Document Organization Overview

This page is intentionally left blank