ODVA™

COMMON INDUSTRIAL PROTOCOL (CIP™) AND THE FAMILY OF CIP NETWORKS

PUB00123R1   FEBRUARY 2016

The Common Industrial Protocol (CIP™) and the Family of CIP Networks

Publication Number: PUB00123R1

PUBLISHER:
ODVA, Inc.

AUTHOR:
Viktor Schiffer – on behalf of Rockwell Automation

EDITORS:
David J. VanGompel – Member of Technical Review Board, ODVA
Raymond A. Romito – Senior Project Engineer, Rockwell Automation
Katherine Voss – Executive Director, ODVA

ABOUT THE AUTHOR
Viktor Schiffer has over 25 years of experience in the field of manufacturing automation and industrial networks. Mr. Schiffer graduated from RWTH Aachen in Germany with a degree of Diplom-Ingenieur in Electrical Engineering and earned a Master of Science degree in Electronic Instrumentation from the University College of Wales in Swansea in the United Kingdom. He served as Engineering Manager for Rockwell Automation in Germany where he was responsible for third party support of the open CIP Networking technologies DeviceNet, ControlNet and EtherNet/IP.

The author has been writing on the topic of industrial networks and has published a number of papers related to the CIP Networking technologies, including specifically on the use of Electronic Data Sheets and on real-time behavior of networks.

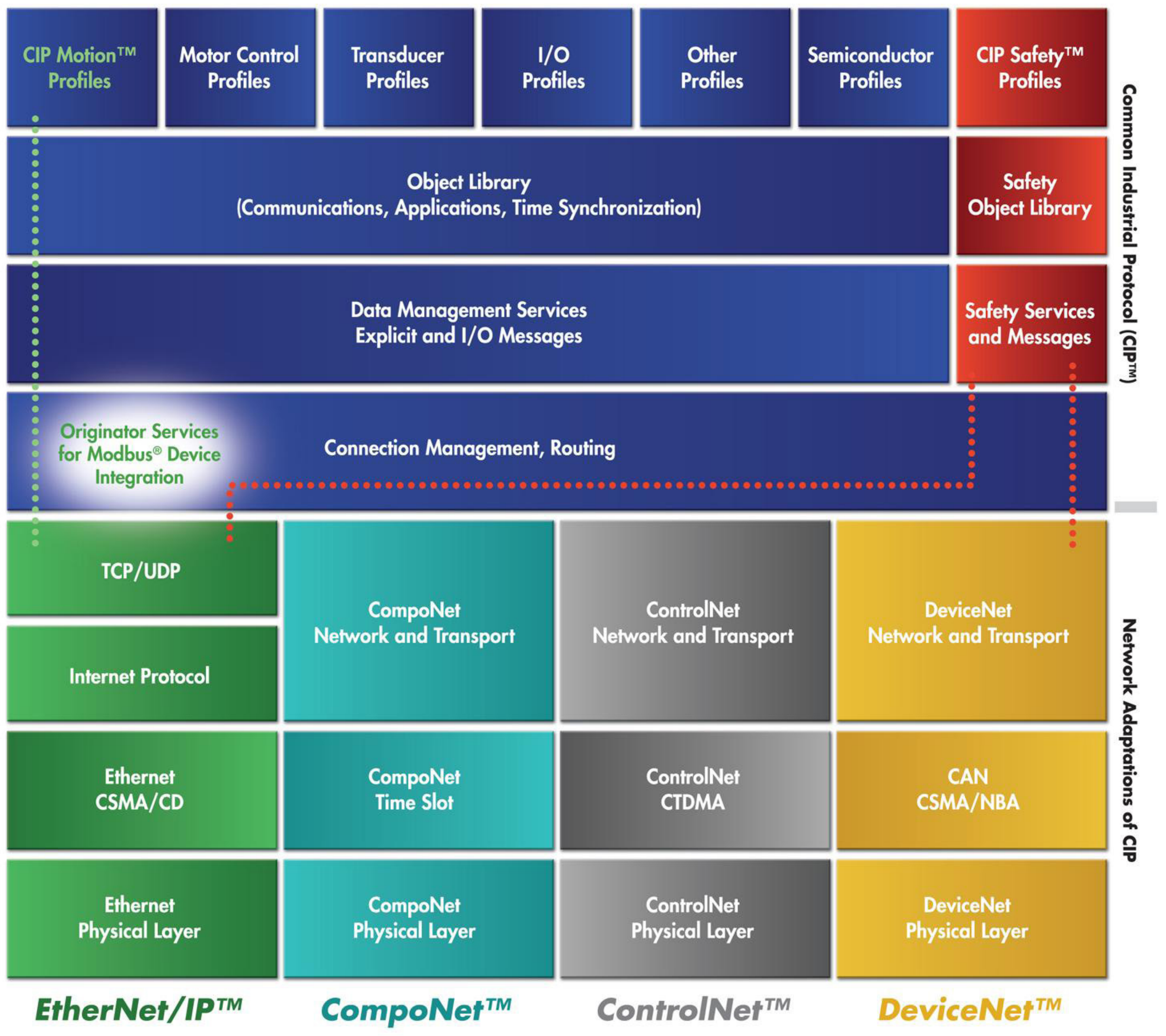
The Common Industrial Protocol (CIP™) and the Family of CIP Networks

Contents

The Common Industrial Protocol (CIP™) and the Family of CIP Networks

Preface

Founded in 1995, ODVA (www.odva.org) is a global association whose members comprise the world's leading automation companies. Combined with the support of its members, ODVA's mission is to advance open, interoperable information and communication technologies in industrial automation. The basis of the membership community is its primary common interest in developing standards and promoting adoption of the Common Industrial Protocol (CIP™), ODVA's media independent network protocol, and the network adaptations of CIP – EtherNet/IP™, DeviceNet™, CompoNet™ and ControlNet™. ODVA manages these technologies, and develops and distributes the specifications for these four networks in a common structure to help ensure consistency and accuracy. The following diagram illustrates the organization of the library of four networks.


CIP Motion™ Profiles
Motor Control Profiles
Transducer Profiles
I/O Profiles
Other Profiles
Semiconductor Profiles
CIP Safety™ Profiles
Object Library (Communications, Applications, Time Synchronization)
Safety Object Library
Data Management Services Explicit and I/O Messages
Safety Services and Messages
Originator Services for Modbus® Device Integration
Connection Management, Routing
Common Industrial Protocol (CIP™)
TCP/UDP
Internet Protocol
Ethernet CSMA/CD
Ethernet Physical Layer
CompoNet Network and Transport
CompoNet Time Slot
CompoNet Physical Layer
ControlNet Network and Transport
ControlNet CTDMA
ControlNet Physical Layer
DeviceNet Network and Transport
CAN CSMA/NBA
DeviceNet Physical Layer
Network Adaptations of CIP
EtherNet/IP™
CompoNet™
ControlNet™
DeviceNet™

Organization of the CIP Networks Specifications

This common structure presents CIP in one volume with a separate volume for each network adaptation of CIP. The specifications for the CIP Networks are typically three-volume sets, structured as shown below and as of April 2015.

The EtherNet/IP Specification consists of:
- Volume 1: Common Industrial Protocol (CIP)
- Volume 2: EtherNet/IP Adaptation of CIP
- Volume 7: Integration of Modbus Devices into the CIP Architecture

The DeviceNet Specification consists of:
- Volume 1: Common Industrial Protocol (CIP)
- Volume 3: DeviceNet Adaptation of CIP
- Volume 7: Integration of Modbus Devices into the CIP Architecture

The ControlNet Specification consists of:
- Volume 1: Common Industrial Protocol (CIP)
- Volume 4: ControlNet Adaptation of CIP
- Volume 7: Integration of Modbus Devices into the CIP Architecture

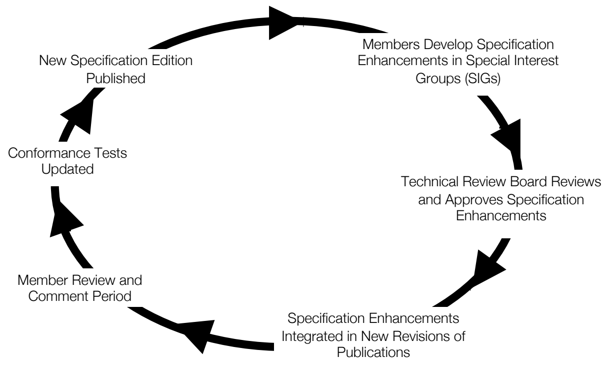
The CIP Safety™ Specification is distributed in a single volume:
- Volume 5: CIP Safety

The CompoNet Specification consists of:
- Volume 1: Common Industrial Protocol (CIP)
- Volume 6: CompoNet Adaptation of CIP
- Volume 7: Integration of Modbus Devices into the CIP Architecture

Chapters 2 of this book provides an overview of the Common Industrial Protocol (CIP). Chapter 3 analyzes the network adaptations of CIP by network, and Chapters 4 and 5 detail the benefits of CIP and the powerful extensions to it, including extensions for synchronization, motion, and functional safety. In conclusion. Chapter 6 outlines the ODVA Conformance Test process, and Chapters 7 through 11 of this book contain reference material (literature references, abbreviations and terminology).

Specification Enhancement Process

The specifications for CIP Networks are continually being enhanced to meet the increasing needs of users for features and functionality. ODVA uses a Specification Enhancement Process in order to ensure open and stable specifications for all CIP Networks. This process is ongoing throughout the year for each CIP Network Specification as shown in the figure below. New editions of each ODVA specification are published on a periodic basis. In recent years, the CIP Networks Library subscriptions have been updated twice annually.


Members Develop Specification Enhancements in Special Interest Groups (SIGs)
Technical Review Board Reviews and Approves Specification Enhancements
Specification Enhancements Integrated in New Revisions of Publications
Member Review and Comment Period
Conformance Tests Updated
New Specification Edition Published

The Common Industrial Protocol (CIP™) and the Family of CIP Networks

1 Introduction

Traditionally, networks used in manufacturing enterprises were optimized for performance in specific applications, most commonly for control, information and safety. While well suited to the functionality for which they were designed, these networks were not developed with a single, coherent enterprise architecture in mind. Since efficiency, reliability and, ultimately, profitability are generally dependent on having more than one of these capabilities, manufacturers were forced to implement several different networks, none of which communicated innately with the other. As a result, over the course of time, most manufacturing enterprise network environments have been characterized by numerous specialized – and generally incompatible – networks existing in one space.

Today, however, corporate expectations for the manufacturing automation network landscape are dramatically different, thanks to the rapid and ubiquitous adoption of Internet technology. Companies of all sizes, all over the world, are trying to find the best ways to connect the entire enterprise. No longer is control of the manufacturing processes enough: the new manufacturing mandate is to enable users throughout the company to access manufacturing data from any location, at any time, and to integrate this data seamlessly with business information systems.

Due to this adoption and expansion of the use of Internet technologies, a rapidly increasing number of users worldwide have looked to "open" systems as a way to connect their disparate enterprise processes. However, the devices, programs and processes used at the various layers of the seven-layer Open Systems Interconnection (OSI) model have different options, capabilities and standards (or lack of). In general, integrating these networks requires extra resources and programming and even then, gaps between the systems often cannot be fully and seamlessly bridged. Consequently, without a way to seamlessly integrate a network, users compromise their investments and rarely achieve all of the productivity and quality benefits promised by open network technology.

Common application layers are the key to advanced communication and true network integration. ODVA's four best-in-class networks — EtherNet/IP™, DeviceNet™, ControlNet™ and CompoNet™ — all are linked by one of industrial automation's most versatile protocols: the Common Industrial Protocol, known as CIP™. CIP encompasses a comprehensive suite of messages and services for the collection of industrial automation applications — control, safety, energy, synchronization & motion, information and network management. CIP allows users to integrate these applications with enterprise-level Ethernet networks and the Internet. Supported by hundreds of vendors around the world and truly media-independent, CIP provides users with a unified communication architecture throughout the industrial enterprise. CIP allows users to benefit today from the many advantages of open networks and protects their existing automation investments, while providing an extensible and upgradable communication architecture.

With media independence comes choice — the ability to choose the CIP Network best suited for an application. As a single, media-independent platform that is shared by a variety of networking technologies, CIP provides the interoperability and interchangeability that is essential to open networks and open systems. Four network adaptations of CIP are available.

EtherNet/IP™: CIP on Ethernet Technology

EtherNet/IP provides users with the network tools to deploy standard Ethernet technology (IEEE 802.3 combined with the TCP/IP Suite) for industrial automation applications while enabling Internet and enterprise connectivity resulting in data anytime and anywhere. EtherNet/IP offers various topology options including a conventional star with standard Ethernet infrastructure devices, or Device Level Ring (DLR) with EtherNet/IP devices so enabled. QuickConnect™ functionality allows devices to be reconnected quickly while the network is running by using an abbreviated start-up procedure.

DeviceNet™: CIP on CAN Technology

DeviceNet provides users with a cost-effective network to distribute and manage simple devices throughout their architecture. DeviceNet uses a trunkline/dropline topology and has DC power available on the network cable to simplify installations by providing a single connection point for network communications and device power up to 24 Vdc, 8 Amps. QuickConnect functionality allows devices to be reconnected quickly while the network is running by using an abbreviated start-up procedure.

ControlNet™: CIP on CTDMA Technology

ControlNet provides users with the tools to achieve deterministic, high-speed transport of time-critical I/O and peer-to-peer interlocks. ControlNet offers a choice of topology options including trunkline/dropline, star or tree. Hardware options are also offered for applications requiring intrinsically safe hardware. Redundant network communication is also available.

CompoNet™: CIP on TDMA Technology

CompoNet enables users to maximize network throughput for applications needing to transmit small packets of data quickly between controllers, sensors and actuators. Its simple network connector and cabling scheme reduces overall system cost and time.

In addition to these network implementations, ODVA has published extensions to CIP for critical applications.

Functional Safety

Safety application coverage in CIP provides the ability to mix safety devices and standard devices on the same network or wire for seamless integration and increased flexibility. CIP Safety™ provides fail-safe communication between nodes such as safety I/O blocks, safety interlock switches, safety light curtains and safety PLCs in safety applications up to Safety Integrity Level (SIL) 3 according to IEC 61508 standards. CIP Safety has also been adopted by Sercos International. A more detailed description of the CIP Safety extension is given in Section 5.2.

Synchronization

Synchronization services in CIP provide the increased control coordination needed for control applications where absolute time synchronization is vital to achieve real-time synchronization between distributed intelligent devices and systems. CIP Sync™ is compliant with IEEE-1588™ standard, and allows synchronization accuracy between two devices of better than 100 nanoseconds. Real-time synchronization can be achieved over conventional 100Mbps, Ethernet systems with a switch-based architecture. A more detailed description of the CIP Sync extension is given in Section 5.1.

Distributed Motion Control

Motion application coverage in CIP eliminates the need for a purpose-built motion-optimized network by allowing high performance motion control and other devices to be combined on a single EtherNet/IP network. This approach results in a modular and streamlined approach to system design and lowers overall system and training cost. CIP Motion™ achieves real-time deterministic behavior of multiple axes through a common sense of time, allowing for 100 axes to be coordinated with a 1 millisecond network update to all axes. Clock synchronization between axes of better than 100 nanoseconds can be readily achieved, meeting the needs of the most demanding motion control applications.

Energy Optimization

Energy application coverage in CIP provides a family of objects and services for the optimization of energy usage (OEU™) and allows scalability of implementation within the device from basic energy awareness to more advanced functions for control of energy, aggregation and reporting of energy information or dynamic demand-response. Further, the CIP family of energy objects and services will allow systems to monitor energy usage and manage energy for efficient energy consumption through dynamic control of energy state and analysis of energy information. Protocol-neutral energy attributes allow for flexibility in the propagation of energy information via multiple protocols to facilitate an e-business model such as capturing energy requirements as a line item on production bills of material or to implement demand-response mechanisms for dynamic energy transactions.

The universal principles of CIP easily lend themselves to possible future implementations on new physical/data link layers. The overall relationship between the four implementations of CIP is shown in Figure 1.

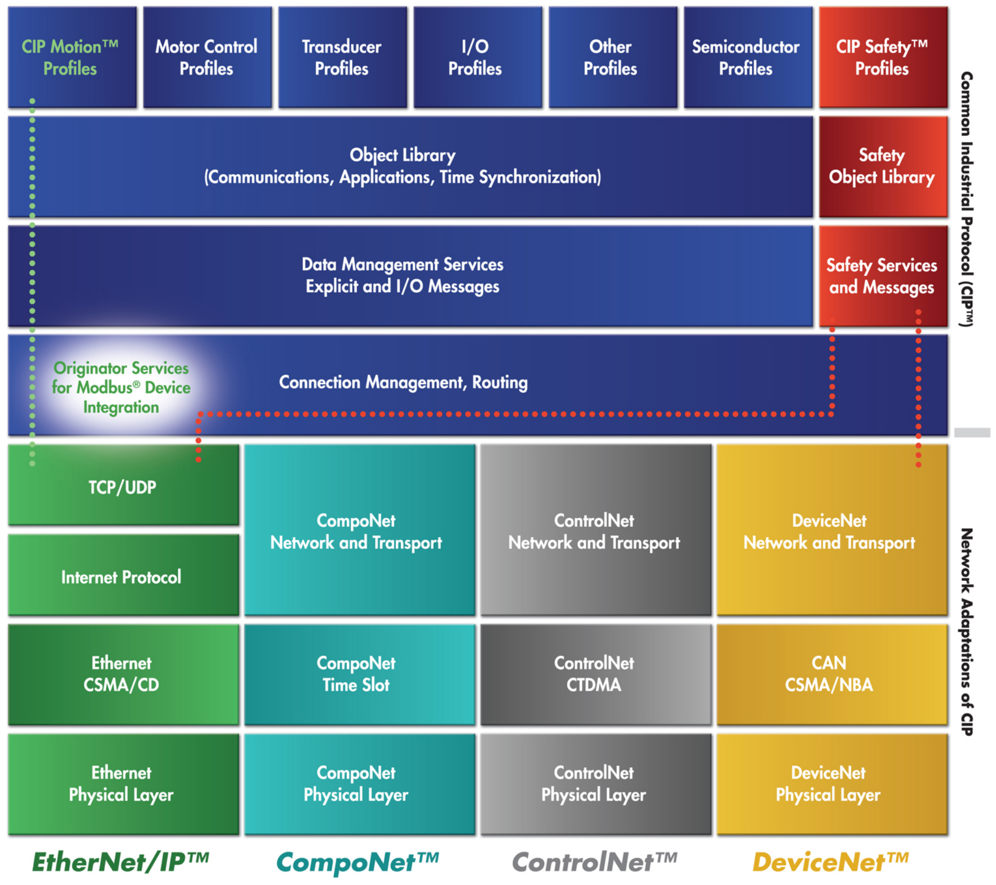

CIP Motion™ Profiles
Motor Control Profiles
Transducer Profiles
I/O Profiles
Other Profiles
Semiconductor Profiles
CIP Safety™ Profiles
Object Library (Communications, Applications, Time Synchronization)
Safety Object Library
Data Management Services Explicit and I/O Messages
Safety Services and Messages
Originator Services for Modbus® Device Integration
Connection Management, Routing
TCP/UDP
Internet Protocol
CompoNet Network and Transport
ControlNet Network and Transport
DeviceNet Network and Transport
Ethernet CSMA/CD
CompoNet Time Slot
ControlNet CTDMA
CAN CSMA/NBA
Ethernet Physical Layer
CompoNet Physical Layer
ControlNet Physical Layer
DeviceNet Physical Layer
EtherNet/IP™
CompoNet™
ControlNet™
DeviceNet™
Common Industrial Protocol (CIP™)
Network Adaptations of CIP

Figure 1 The Common Industrial Protocol and its network adaptations

2 Description of the CIP Networks Library

CIP is a very versatile protocol designed with the automation industry in mind. However, due to its open nature, it can be and has been applied to many more areas. The CIP Networks Library contains several volumes:

- **Volume 1** deals with the common aspects of CIP that apply to all of the network adaptations. This volume contains the common object library and the device profile library, along with a general description of the communications model, device configuration and CIP data management. This volume also defines an auxiliary power distribution system that is common to all adaptations of CIP.

- **Volume 2** is the EtherNet/IP Adaptation of CIP, which describes how CIP is adapted to the Ethernet TCP/IP and UDP/IP transportation layers. It also contains any extensions to the material in Volume 1 that are necessary for EtherNet/IP, such as the optional industrial physical layer and connectors.

- **Volume 3** is the DeviceNet Adaptation of CIP, which describes how CIP is adapted to the CAN data link layer. It also contains any extensions to the material in Volume 1 that are necessary for DeviceNet.

- **Volume 4** is the ControlNet Adaptation of CIP, which describes how CIP is adapted to the ControlNet data link layer. It contains a complete description of the ControlNet data link layer and any extensions to the material in Volume 1 that are necessary for ControlNet.

- **Volume 5** is CIP Safety. It contains the information necessary to implement the CIP Safety protocol on CIP Networks.

- **Volume 6** is the CompoNet Adaptation of CIP, which describes how CIP is adapted to the CompoNet data link layer. It contains a complete description of the CompoNet data link layer and any extensions to the material in Volume 1 that are necessary for CompoNet.

- **Volume 7** is the Integration of Modbus Devices into the CIP Architecture. This volume describes a standard for the integration of Modbus devices into the CIP world.

For brevity, this document will use the volume numbers above when referencing the different books in the CIP Networks Library.

Specifications for the CIP Networks referenced above, and other documents discussing CIP, are available from ODVA at www.odva.org. It is beyond the scope of this book to fully describe each and every detail of CIP, but key features of the protocol and the auxiliary power distribution system will be discussed, including:

- Object modeling;
- Services;
- Messaging protocol;
- Communication objects;
- Object library;
- Device profiles;
- Configuration and electronic data sheets;
- Bridging and routing;
- Data management;
- Auxiliary power distribution system.

A few terms used throughout this section are described here to make sure they are well understood, further terms are described in section 11:

- **Client:**
Within a client/server model, the client is the device that sends a request to a server. The client expects a response from the server.

- **Server:**
Within a client/server model, the server is the device that receives a request from a client. The server is expected to give a response to the client.

- **Producer:**
Within the producer/consumer model, the producing device places a message on the network for consumption by one or several consumers. Generally, the produced message is not directed to a specific consumer.

- **Consumer:**
Within the producer/consumer model, the consumer is one of potentially several consuming devices that picks up a message placed on the network by a producing device.

- **Producer/Consumer Model:**
The producer/consumer model is inherently multicast. Nodes on the network determine if they should consume the data in a message based on the connection ID in the packet. CIP uses the producer/consumer model, as opposed to the traditional source/destination message addressing scheme (see Figure 2).

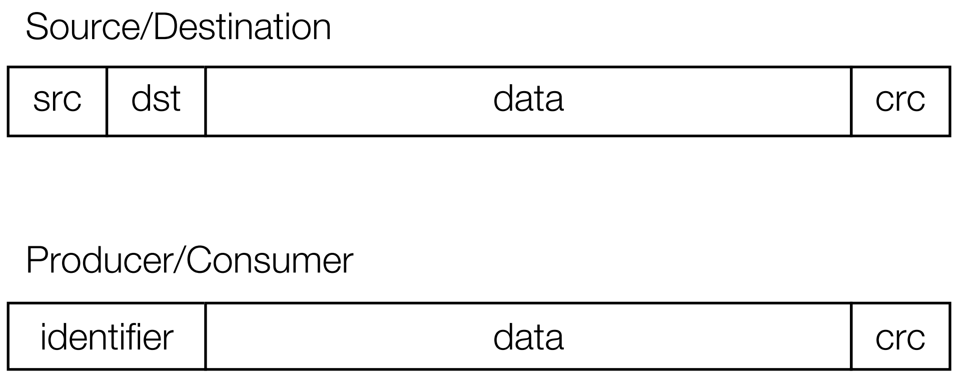


Figure 2 Source/Destination vs. Producer/Consumer Model

- **Explicit Message:**
Explicit messages contain addressing and service information that directs the receiving device to perform a certain service (action) on a specific part (e.g., an attribute of a given object) of a device.

- **Implicit (I/O) Message:**
Implicit messages do not carry address and/or service information; any consuming nodes already know what to do with the data based on the connection ID that was assigned when the connection was established. Implicit messages are so named because the meaning of the data is "implied" by the connection ID. In most cases they are used to transport I/O data.

Let's have a look at the individual elements of CIP:

2.1. Object Modeling

CIP uses abstract object modeling to describe:

- The suite of available communication services;

- The externally visible behavior of a CIP node;

- A common means by which information within CIP products is accessed and exchanged.

Every CIP node is modeled as a collection of objects. An object provides an abstract representation of a particular component within a product. Anything not described in object form is not visible through CIP. CIP objects are structured into classes, instances and attributes.

A class is a set of objects that all represent the same kind of system component. An object instance is the actual representation of a particular object within a class. Each instance of a class has the same attributes, but also has its own particular set of attribute values. As Figure 3 illustrates, multiple object instances within a particular class can reside within a CIP node.

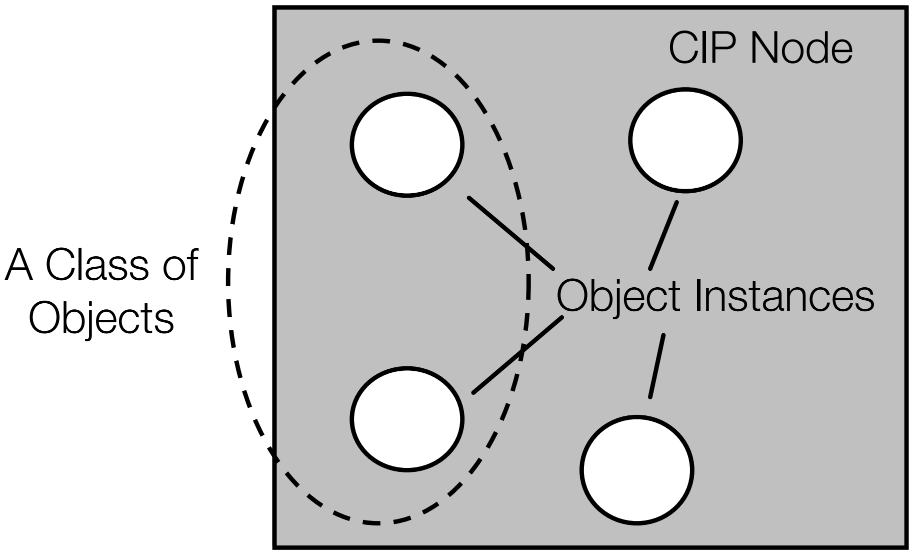


Figure 3 A Class of Objects

In addition to the instance attributes, an object class may also have class attributes. These are attributes that describe properties of the whole object class, e.g., how many instances of this particular object exist. Furthermore, both object instances and the class itself exhibit a certain behavior and allow certain services to be applied to the attributes, instances or to the whole class. All publicly defined objects that are implemented in a device must follow at least the mandatory requirements defined in the various CIP Networks specifications. Vendor-specific objects may also be defined with a set of instances, attributes and services according to the requirements of the vendor. However, they need to follow certain rules that are also set forth in the specifications. The objects and their components are addressed by a uniform addressing scheme consisting of:

- **Node Address:**
An integer identification value assigned to each node on a CIP Network. On DeviceNet, ControlNet and CompoNet, this is also called a MAC ID (Media Access Control Identifier) and is nothing more than the node number of the device. On EtherNet/IP, the node address is the IP address.

- **Class Identifier (Class ID):**
An integer identification value assigned to each object class accessible from the network.

- **Instance Identifier (Instance ID):**
An integer identification value assigned to an object instance that identifies it among all instances of the same class.

- **Attribute Identifier (Attribute ID):**
An integer identification value assigned to a class or instance attribute.

- **Service Code:**
An integer identification value which denotes an action request that can be directed at a particular object instance or object attribute (see Section 2.2).

Object class identifiers are divided into two types of objects: publicly defined objects (ranging from 0x0000 – 0x0063 and 0x00F0 – 0x02FF) and vendor-specific objects (ranging from 0x0064 – 0x00C7 and 0x0300 – 0x04FF). All other class identifiers are reserved for future use. In some cases, e.g., within the assembly object class, instance identifiers are divided into two types of instances: publicly defined (ranging from 0x0001 – 0x0063 and 0x00C8 – 0x02FF) and vendor-specific (ranging from 0x0064 – 0x00C7 and 0x0300 – 0x04FF). All other instance identifiers are reserved for future use. Attribute identifiers are divided into two types of attributes: publicly defined (ranging from 0x0000 – 0x0063, 0x0100 – 0x02FF and 0x0500 – 0x08FF) and vendor-specific (ranging from 0x0064 – 0x00C7, 0x0300 – 0x04FF and 0x0900 – 0x0CFF). All other attribute identifiers are reserved for future use. While vendor-specific objects can be created with a great deal of flexibility, these objects must adhere to certain rules specified for CIP, e.g., they can use whatever instance and attribute IDs the developer wishes, but their class attributes must follow guidelines detailed in Volume 1, Chapter 4 of the CIP Networks Library.

Addressing objects and their attributes can be performed with 8 bit, 16 bit or 32 bit addresses. In most cases, class and instance addresses are 8 or 16 bit wide, and attribute addresses are only 8 bit wide. 32 bit addresses are currently reserved for instance addressing only.

Figure 4 shows an example of this object addressing scheme.

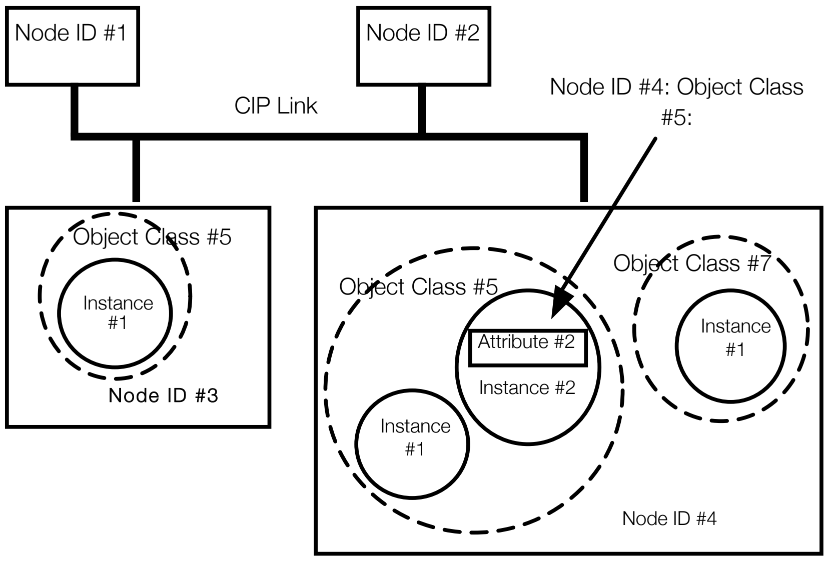


Figure 4 Object Addressing Example

2.2. Services

Service codes are used to define the action that is requested to take place when an object or parts of an object are addressed through explicit messages using the addressing scheme described in Section 2.1. Apart from simple read and write functions, a set of CIP services has been defined. These CIP services are common in nature, meaning they may be used in all CIP Networks and they are useful for a variety of objects. Furthermore, there are object-specific service codes that may have a different meaning for the same code, depending on the class of object. Finally, defining vendor-specific services according to the requirements of the product developer is possible. While this provides a lot of flexibility, the disadvantage of vendor-specific services is that they may not be understood universally. Minimally, vendors provide a description of the public information that their customers will need access to in their literature.

2.3. Messaging Protocol

CIP is a connection-based protocol. A CIP connection provides a path between multiple application objects. When a connection is established, the transmissions associated with that connection are assigned a connection ID, or CID (see Figure 5). If the connection involves a bi-directional exchange, then two CID values are assigned.

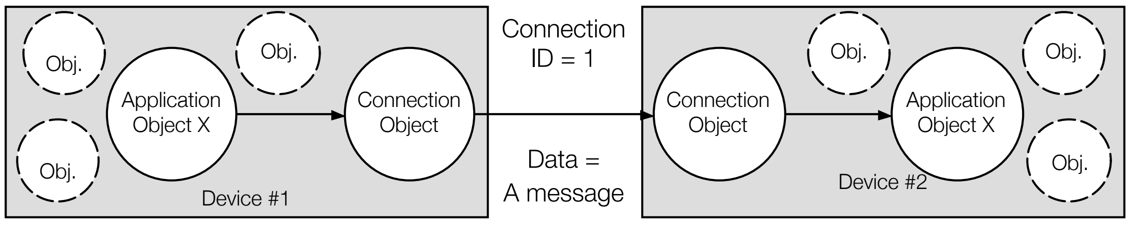


Figure 5 Connections and Connection IDs

The definition and format of the CID is network dependent. For example, the CID for CIP connections over DeviceNet is based on the CAN identifier field.

Since most messaging on a CIP Network is done through connections, a process has been defined to establish such connections between devices that are not yet connected. This is done through the unconnected message manager (UCMM) function, which is responsible for processing unconnected explicit requests and responses.

Establishing a CIP connection is generally accomplished by sending a UCMM Forward_Open service request message. The Forward_Open is required for all devices that support connections on ControlNet and EtherNet/IP. CompoNet uses a different method described in Section 3.4.12. DeviceNet only uses the simplified methods described in Sections 3.1.16 and 3.1.17.

A Forward_Open request contains all information required to create a connection between the originator and the target device. The resulting data exchange may be unidirectional or bidirectional. In particular, the Forward_Open request contains information on the following:

- Time-out information for this connection;
- Network CID for the connection from the originator to the target;
- Network CID for the connection from the target to the originator;
- Information about the identity of the originator (vendor ID and serial number);
- Maximum data sizes of the messages on this connection;
- Whether it will be unicast or multicast;
- Trigger mechanisms, e.g., cyclic, change of state (COS);
- Electronic key so the target node can verify that it is the proper type of node (optional);
- Connection path for the application object data in the node that will be produced and consumed;
- Data Segment containing configuration information for the node (optional);
- Routing information if the connection is to span more than one network (optional).

Some networks, like ControlNet, EtherNet/IP and CompoNet, also make use of unconnected explicit messaging. DeviceNet uses unconnected explicit messaging only to establish connections.

All connections on a CIP Network can be categorized as I/O connections or explicit messaging connections.

- I/O connections provide dedicated, special-purpose communication paths between a producing application and one or more consuming applications. Application-specific I/O data move through these ports, a process that is often referred to as implicit messaging. These messages can be unicast or multicast. These connections are also called "implicit" connections because the meaning of the data is implied by the connection ID.

- Explicit messaging connections provide generic, multi-purpose communication paths between two devices. These connections often are referred to simply as messaging connections. Explicit messages provide typical request/response-oriented network communications. These messages are point-to-point. They are called "explicit" messages because the data in the request explicitly states what service and object is being requested.

The actual data transmitted in CIP I/O messages are the I/O data in an appropriate format; e.g., the data may be prefixed by a sequence count value. This sequence count value can be used to distinguish old data from new, e.g., if a message has been re-sent as a heartbeat in a COS connection. The two states "Run" or "Idle" can be indicated with an I/O message either by prefixing a real-time header, as is primarily used for ControlNet and EtherNet/IP, or by sending I/O data (Run) or no I/O data (Idle), a process primarily used for DeviceNet. CompoNet uses a bit within the OUT frame or the TRG frame to indicate the states "Run" and "Idle". "Run" is the normal operational state of a device with the outputs under the control of the controlling application, while the reaction to receiving an "Idle" event is vendor-specific and application-specific. Typically, this means bringing all outputs of the device to a predefined, safe "Idle" state (which usually means "off"), i.e., de-energized.

Explicit messaging requests contain a service code with path information to the desired object within the target device followed by data (if any). The associated responses repeat the service code followed by status fields followed by data (if any). DeviceNet and CompoNet use a "condensed" format for explicit messages in most cases, while ControlNet and EtherNet/IP only use the "full" format.

2.4. Communication Objects

CIP communication objects manage and provide the run-time exchange of messages. Communication objects are unique in that they are the focal points for all CIP communication. It therefore makes sense to look at them in more detail.

Each communication object contains a link producer part, a link consumer part or both. I/O connections may be either producing or consuming or producing and consuming, while explicit messaging connections are always producing and consuming.

Figure 6 and Figure 7 show the typical connection arrangement for CIP I/O messaging and CIP explicit messaging. The attribute values in the connection objects define a set of attributes that describe vital parameters of this connection. Note that explicit messages are always directed to the message router object.

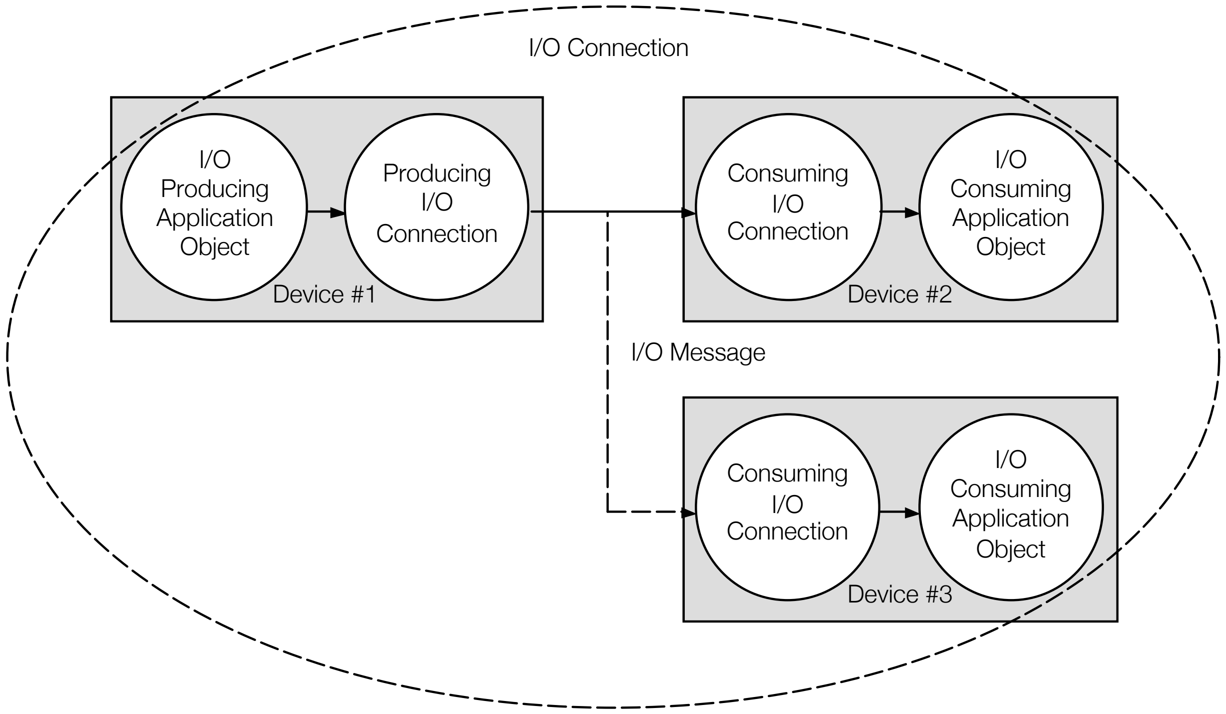


Figure 6 CIP Multicast I/O Connection

The attribute values of a connection object specify whether it is an I/O connection or an explicit messaging connection, the maximum size of the data to be exchanged across this connection and the source and destination of the data. Further attributes define the state and behavior of the connection. Particularly important behaviors include how messages are triggered (from the application, through change of state (COS), i.e., when data has changed, through cyclic events or by network events) and the timing of the connections (time-out associated with this connection and predefined action if a time-out occurs). CIP allows multiple connections to coexist in a device, although simple devices – e.g., simple DeviceNet or CompoNet slaves – typically will only have one or two live connections at any time (only one connection on CompoNet).

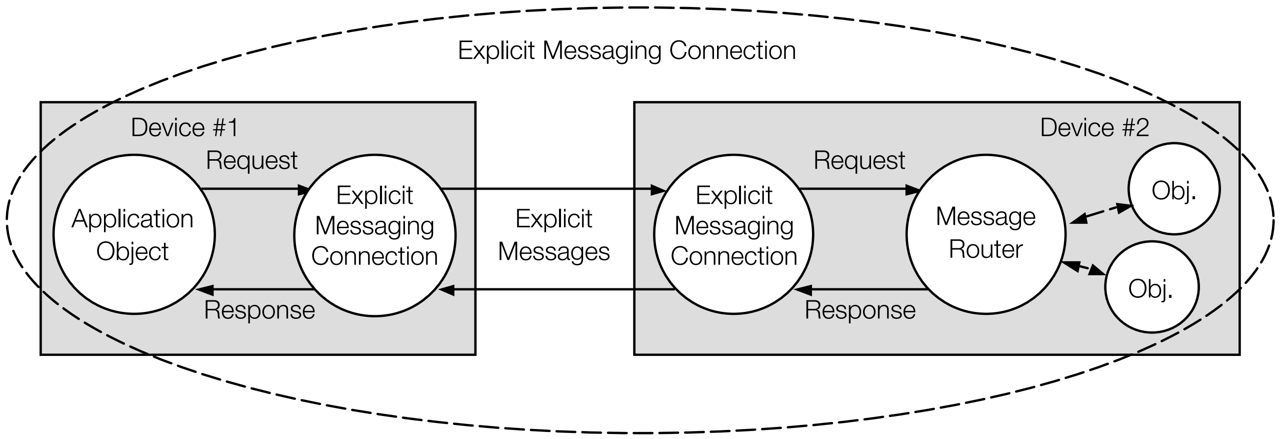


Figure 7 CIP Explicit Messaging Connection

2.5. Object Library

The CIP family of protocols contains a large collection of commonly defined objects. The overall set of object classes can be subdivided into three types:

- General use;
- Application specific;
- Network specific.

Objects defined in Volume 1 of the CIP Networks Library are available for use on all network adaptations of CIP. Some of these objects may require specific changes or limitations when implemented on some of the network adaptations. These exceptions are noted in the network-specific volume. Therefore, to see a complete picture of a specific network implementation of an object, refer to Chapter 5 in both the Protocol Adaptation Volume and Volume 1.

The following are general use objects (object IDs in brackets):

- Assembly (0x04)	- Message Router (0x02)
- Acknowledge Handler (0x2B)	- Originator Connection List (0x45)
- Connection (0x05)	- Parameter (0x0F)
- Connection Configuration (0xF3)	- Parameter Group (0x10)
- Connection Manager (0x06)	- Port (0xF4)
- File (0x37)	- Register (0x07)
- Identity (0x01)	- Selection (0x2E)

The following group of objects is application-specific (object IDs in brackets):

- AC/DC Drive (0x2A)	- Motion Device Axis (0x42)
- Analog Group (0x22)	- Motor Data (0x28)
- Analog Input Group (0x20)	- Non-Electrical Energy (0x50)
- Analog Input Point (0x0A)	- Overload (0x2C)
- Analog Output Group (0x21)	- Position Controller (0x25)
- Analog Output Point (0x0B)	- Position Controller Supervisor (0x24)
- Base Energy (0x4E)	- Position Sensor (0x23)
- Block Sequencer (0x26)	- Power Curtailment Object (0x5C)
- Command Block (0x27)	- Power Management Object (0x53)
- Control Supervisor (0x29)	- Presence Sensing (0x0E)
- Discrete Group (0x1F)	- S-Analog Actuator (0x32)
- Discrete Input Group (0x1D)	- S-Analog Sensor (0x31)
- Discrete Output Group (0x1E)	- S-Device Supervisor (0x30)
- Discrete Input Point (0x08)	- S-Gas Calibration (0x34)
- Discrete Output Point (0x09)	- S-Partial Pressure (0x38)
- Electrical Energy (0x4F)	- S-Sensor Calibration (0x40)
- Event Log (0x41)	- S-Single Stage Controller (0x33)
- Group (0x12)	- Safety Analog Input Group (0x4A)

- Safety Analog Input Point (0x49)
- Safety Dual Channel Feedback Object (0x59)
- Safety Feedback Object (0x5A)
- Safety Discrete Input Group (0x3E)
- Safety Discrete Input Point (0x3D)
- Safety Discrete Output Group (0x3C)
- Safety Discrete Output Point (0x3B)
- Safety Dual Channel Analog Input (0x4B)
- Safety Dual Channel Output (0x3F)
- Safety Limit Functions Object (0x5B
- Safety Stop Functions Object (0x5A)
- Safety Supervisor (0x39)
- Safety Validator (0x3A)
- Softstart (0x2D)
- Target Connection List (0x4D)
- Time Sync (0x43)
- Trip Point (0x35)

The last group of objects is network-specific (object IDs in brackets):

- Base Switch (0x51)
- CompoNet Link (0xF7)
- CompoNet Repeater (0xF8)
- ControlNet (0xF0)
- ControlNet Keeper (0xF1)
- ControlNet Scheduling (0xF2)
- Device Level Ring (DLR) (0x47)
- DeviceNet (0x03)
- Ethernet Link (0xF6)
- Modbus (0x44)
- Modbus Serial Link (0x46)
- Parallel Redundancy Protocol (0x56)
- Power Management (0x53)
- PRP Nodes Table (0x57)
- SERCOS III Link (0x4C)
- SNMP (0x52)
- QoS (0x48)
- RSTP Bridge (0x54)
- RSTP Port (0x55)
- TCP/IP Interface (0xF5)

The general use objects can be found in many different devices, while the application-specific objects are typically found only in devices hosting such applications. New objects are added on an ongoing basis by the various ODVA Special Interest Groups (SIGs). As mentioned earlier, there are many vendor-specific objects defined by developers to satisfy needs that may not be met by the existing open objects contained in the published specifications.

Although this looks like a large number of object types, typical devices implement only a subset of these objects. Figure 8 shows the object model of such a typical device.

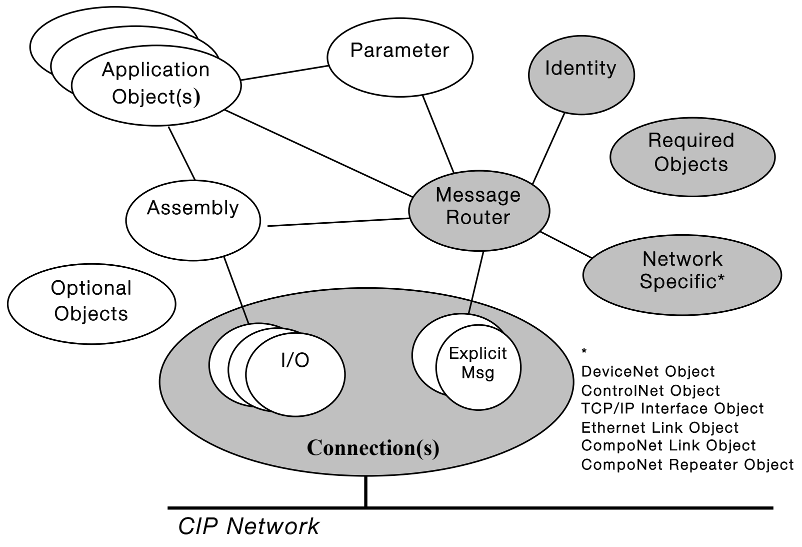


Figure 8 Typical Device Object Model

The objects required in a typical device are:

- Either a connection object or a connection manager object;
- An identity object;
- One or several network-specific link objects (depends on network);
- A Message router object (at least its function).

Further objects are added according to the functionality of the device. This enables scalability for each implementation so that simple devices, such as proximity sensors, are not burdened with unnecessary overhead. Developers typically use publicly defined objects (see above list), but can also create their own objects in the vendor-specific areas, e.g., Class ID 100 – 199. However, they are strongly encouraged to work with the Special Interest Groups (SIGs) of ODVA to create common definitions for additional objects instead of inventing private ones.

Out of the general use objects, several are described in more detail below.

2.5.1. Identity Object (Class ID: 0x01)

The identity object is described in greater detail because, being a relatively simple object, it can serve to illustrate the general principles of CIP objects. In addition, every device must have an identity object.

The vast majority of devices support only one instance of the identity object. Thus, typically there are no requirements for any class attributes that describe additional class details, e.g., how many instances exist in the device. Only instance attributes are required in most cases. These are as follows:

Mandatory Attributes:

- Vendor ID	- Status
- Device Type	- Serial Number
- Product Code	- Product Name
- Revision	

Optional or Conditional Attributes:

- State	- Semaphore
- Configuration Consistency Value	- Assigned_Name
- Heartbeat Interval	- Assigned_Description
- Active Language	- Geographic_Location
- Supported Language List	- Modbus Identity Info
- International Product Name	- Protection Mode

Let us have a look at these attributes in more detail:

- The **Vendor ID** attribute identifies the vendor that markets the device. This UINT (Unsigned Integer) value (for Data Type descriptions see Section 2.9) is assigned to a specific vendor by ODVA. If a vendor intends to build products for more than one CIP Network, the same Vendor ID will generally be assigned for all networks, but they must be registered independently with ODVA prior to use.

- The **Device Type**, again a UINT value, specifies which profile has been used for this device. It must be one of the vendor IDs listed in Volume 1, Chapter 6 of the CIP Networks Library or a vendor-specific type (see Section 2.6).

- The **Product Code** is a UINT number defined by the vendor of the device. This code is used to distinguish multiple products of the same vendor ID from the same vendor. There is generally a loose association between a Catalog Number and aproduct code, but not necessarily.

- The **Revision** is split into two USINT (Unsigned Short Integer) values specifying a major revision and a minor revision. Any device change(s) that result in modifying the behavior of the device on the network must be reflected in a change to the minor revision at minimum. Any changes in the device's logical interface, e.g., additional attributes, modified/additional I/O assemblies etc., require a change to the major revision and this change must be reflected in a revised Electronic Data Sheet (EDS) (see Section 2.7). vendor ID, vendor ID,product code and major revision provide an unambiguous identification of an EDS for this device.

- The **Status** attribute provides information on the status of the device, e.g., whether it is owned (controlled by another device) or configured (to something different than the out-of-the-box default), and whether any major or minor faults have occurred.

- The **Serial Number** is used to uniquely identify individual devices in conjunction with the vendor ID, i.e., no two CIP devices from one vendor may carry the same serial number. The 32 bits of the serial number allow ample space for a subdivision into number ranges that can be used by different divisions of larger companies.

- The **Product Name** attribute allows the vendor to give a meaningful ASCII name string (up to 32 characters) to the device.

- The **State** attribute describes the state of a device in a single UINT value. It is less detailed than the Status attribute.

- The **Configuration Consistency Value** allows a distinction between a device that has been configured and one that has not, or between different configurations in a single device. This helps avoid unnecessary configuration downloads.

- The **Heartbeat Interval** enables the Device Heartbeat Message. This is an unconnected change-of-state message that has a settable background cyclic interval between one and 255 seconds. Currently, this option is only defined for use on DeviceNet.

- The **Supported Language List** and **International Product Name** attributes can be used to describe the product in multiple languages and the **Active Language** attribute then specifies which of the supported languages is in use.

- The **Semaphore** attribute provides a semaphore for client access synchronization to the entire device.

- The **Assigned_Name, Assigned_Description** and **Geographical_Location** attributes can be used to individualize products by the user of the product.

- The **Modbus Identity Info** attribute can provide identity information in Modbus format to the extent the device supports it.

- The **Protection Mode** attribute is an indication of the present mode of protection for the device.

The services supported by the class and instance attributes are either Get_Attribute_Single (typically implemented in DeviceNet and CompoNet devices) or Get_Attributes_All (typically implemented in ControlNet and EtherNet/IP devices). The only attributes that can be set are: the Heartbeat Interval, the Active Language, the Semaphore, the Assigned_Name, Assigned_Description and Geographical_Location attributes. The only other service that typically is supported by the identity object is the Reset service. This Reset service comes with three different options that can let the device restart in three different ways.

The behavior of the identity object is described through a state transition diagram.

2.5.2. Parameter Object (Class ID: 0x0F)

This object is described in some detail since it is referred to in Section 2.7, Configuration and Electronic Data Sheets. When used, the parameter object comes in two types: a complete object and an abbreviated version (parameter object Stub). This abbreviated version is used primarily by DeviceNet and CompoNet devices that have only small amounts of memory available. The parameter object stub, in conjunction with the Electronic Data Sheet, has roughly the same functionality as the full object (see Section 2.7).

The purpose of the parameter object is to provide a general means of allowing access to many attributes of the various objects in the device without requiring a tool (such as a handheld terminal) to have any knowledge about specific objects in the device.

The class attributes of the parameter object contain information about how many instances exist in this device and a Class Descriptor, indicating, among other properties, whether a full or a stub version is supported. In addition, class attributes tell whether a configuration assembly is used and what language is used in the parameter object.

The first six instance attributes are required for the object stub. These are:

Parameter Value	The actual parameter
Link Path Size	These two attributes describe the application object/ instance/ attribute from which the parameter value was retrieved.
Descriptor	This describes parameter properties, e.g., read-only, monitor parameter, etc.
Data Type	This describes the data type (e.g., size, range) using a standard mechanism defined by CIP
Data Size	Data size in bytes

These six attributes allow access, interpretation and modification of the parameter value, but the remaining attributes make it much easier to understand the meaning of the parameter:

- The next three attributes provide ASCII strings with the name of the parameter, its engineering units and an associated help text;
- Another three attributes contain the minimum, maximum and default values of the parameter;
- Four more attributes can link the scaling of the parameter value so that the parameter can be displayed in a more meaningful way, e.g., raw value in multiples of 10 mA, scaled value displayed in amps;
- Another four attributes can link the scaling values to other parameters. This feature allows variable scaling of parameters, e.g., percentage scaling to a full range value that is set by another parameter;
- Attribute #21 defines how many decimal places are to be displayed if the parameter value is scaled;
- Finally, the last three attributes are an international language version of the parameter name, its engineering units and the associated help text.

2.5.3. Assembly Object (Class ID: 0x04)

Assembly objects provide the option of mapping data from attributes of different instances of various classes into one single attribute (#3) of an assembly object. This mapping is generally used for I/O messages to maximize the efficiency of the control data exchange on the network. Assembly mapping makes the I/O data available in one block; thus, there are fewer connection object instances and fewer transmissions on the network. The process data are normally combined from different application objects. An assembly object also can be used to configure a device with a single data block, alleviating the need to set individual parameters.

CIP makes a distinction between input and output assemblies. “input” and “output” in this context are viewed from the perspective of the controlling element (e.g., a PLC/PAC). An input assembly in a device collects data from the input application (e.g., field wiring terminal, proximity sensor, etc.) and produces it on the network, where it is consumed by the controlling device and/or operator interface. An output assembly in a device consumes data that the controlling element sends to the network and writes that data to the output application (e.g., field wiring terminals, motor speed control, etc.). This data mapping is very flexible; even mapping of individual bits is permitted. Assemblies also can be used to transmit a complete set of configurable parameters instead of accessing them individually. These assemblies are called configuration assemblies.

Figure 9 shows an example of assembly mapping. The data from application objects 100 and 101 are mapped in two instances of the assembly object. Instance 1 is set up as an input assembly for the input data, and instance 2 as an output assembly for output data. The data block is always accessed via attribute 3 of the relevant assembly instance. Attributes 1 and 2 contain mapping information.

I/O assembly mapping is specified for many Device Profiles in Chapter 6 of Volume 1. Device developers then can choose which assemblies they support in their products. If none of the publicly defined assemblies fully represents the functionality of the product, a device vendor may implement additional vendor-specific assemblies (Instance IDs 100 – 199).

CIP defines static and dynamic assembly objects. Whereas mapping for static assemblies is permanently programmed in the device (ROM), dynamic assemblies can be modified and extended through dynamic mapping (RAM). Most simple CIP devices support only static assembly objects. Dynamic assembly objects may be used in more complex devices, but they are not very common.

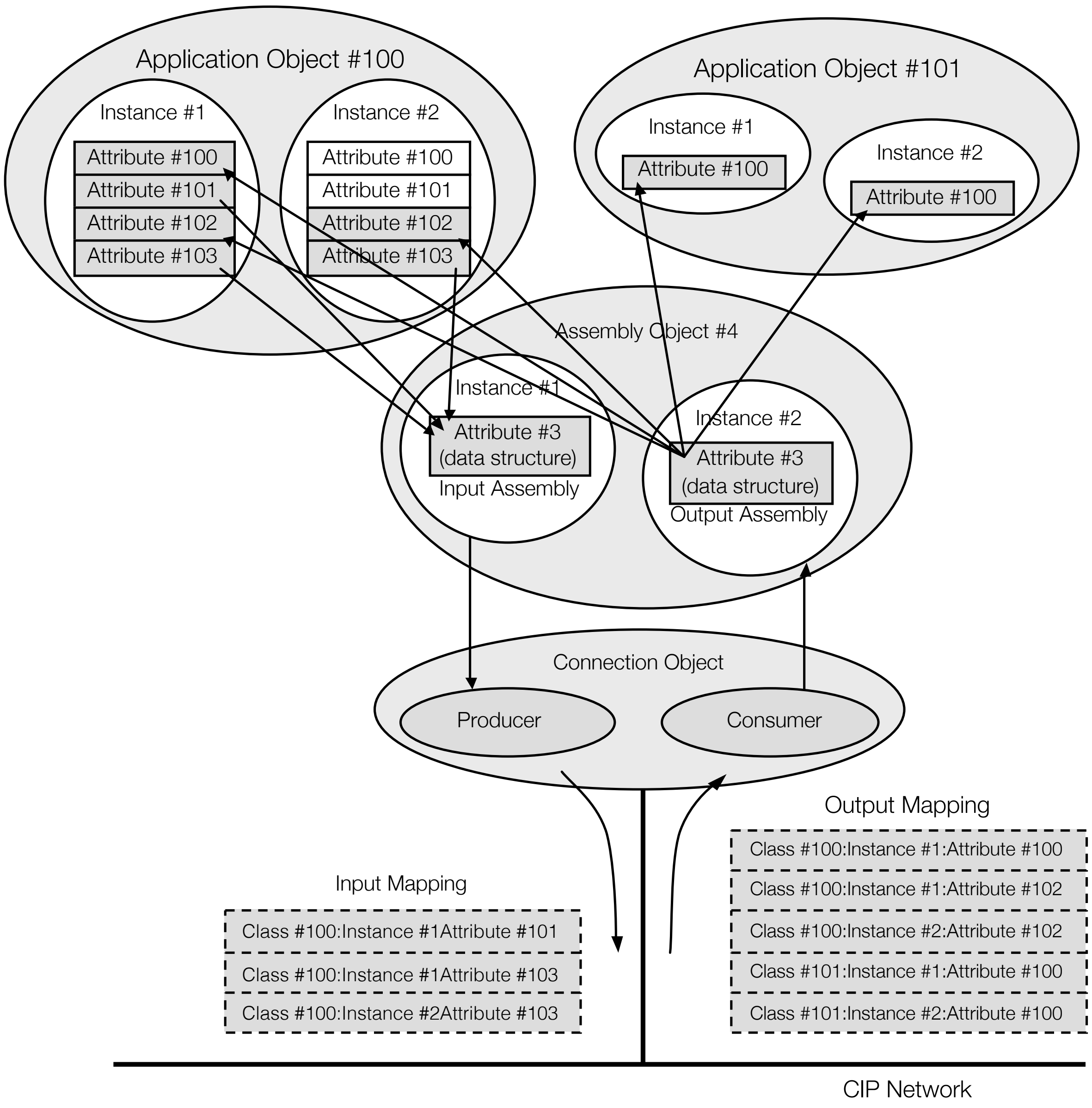


Figure 9 Example of an Assembly Mapping in a Typical I/O Device

2.6. Device Profiles

It would be possible to design products using only the definitions of communication networks and objects, but this could easily result in similar products having quite different data structures and behavior. To overcome this situation and to make the application of CIP devices much easier, devices of similar functionality have been grouped into vendor IDs with associated profiles. Such a CIP profile contains the full description of the object structure and behavior. The following vendor IDs and associated profiles are defined in Volume 1 (profile numbers in parentheses):

- AC Drives (0x02)
- CIP Modbus Device (0x28)
- CIP Modbus Translator (0x29)
- CIP Motion Drive (0x25)
- CIP Motion Encoder (0x2F)
- CIP Motion I/O (0x31)
- CIP Motion Safety Drive Device (0x2D)
- Communications Adapter (0x0C)
- CompoNet Repeater (0x26)
- Contactor (0x15)
- ControlNet Physical Layer Component (0x32)
- DC Drives (0x13)
- DC Power Generator (0x1F)
- Embedded Component (0xC8)
- Encoder (0x22)
- Enhanced Mass Flow Controller (0x27)
- Fluid Flow Controller (0x24)
- General Purpose Discrete I/O (0x07)
- Generic Device, keyable (0x2B)
- Human Machine Interface (HMI) (0x18)
- Inductive Proximity Switch (0x05)
- Limit Switch (0x04)
- Managed Ethernet Switch (0x2C)
- Mass Flow Controller (0x1A)
- Mass Flow Controller, Enhanced (0x27)
- Motor Overload Device (0x03)
- Motor Starter (0x16)
- Photoelectric Sensor (0x06)
- Pneumatic Valve(s) (0x1B)
- Position Controller (0x10)
- Process Control Valve (0x1D)
- Programmable Logic Controller (0x0E)
- Residual Gas Analyzer (0x1E)
- Resolver (0x09)
- RF Power Generator (0x20)
- Safety Analog I/O Device (0x2A)
- Safety Drive Device (0x2E)
- Safety Discrete I/O Device (0x23)
- Softstart Starter (0x17)
- Turbomolecular Vacuum Pump (0x21)
- Vacuum/Pressure Gauge (0x1C)

Device developers must use a vendor ID to uniquely identify their product. Any device that does not fall into the scope of one of the specialized Device Profiles must use the Generic Device profile (0x2B) or a vendor-specific profile. What profile is used and which parts of it are implemented must be described in the user's device documentation.

Every profile consists of a set of objects – some required, some optional – and a behavior associated with that particular type of device. Most profiles also define one or more I/O data formats (assemblies) that define the makeup of the I/O data. In addition to the commonly-defined objects and I/O data assemblies, vendors can add objects and assemblies of their own if their devices provide additional functionality. Furthermore, vendors can create profiles within the vendor-specific profile range. They are then free to define whatever behavior and objects are required for their device as long as they adhere to the general rules for profiles outlined in Chapter 6 of Volume 1 of the CIP Networks Library. Whenever additional functionality is needed by multiple vendors, ODVA encourages coordinating these new features through Special Interest Groups (SIGs), which can then create new profiles or additions to existing profiles for everybody's use and for the benefit of the device users.

All open (ODVA defined) profiles carry numbers in the 0x00 through 0x63 or 0x0100 through 0x02FF ranges, while vendor-specific profiles carry numbers in the 0x64 through 0xC7 or 0x0300 through 0x02FF ranges. All other profile numbers are reserved by CIP.

2.7. Configuration and Electronic Data Sheets

CIP provides several options for configuring devices:

- A printed data sheet;
- Parameter objects and parameter object stubs;
- An Electronic Data Sheet (EDS);
- A combination of an EDS and parameter object stubs;
- A configuration assembly combined with any of the above methods.

When using configuration information collected on a printed data sheet, configuration tools can only provide prompts for service, class, instance and attribute data and relay this information to a device. While this procedure can do the job, it is the least desirable solution since it does not determine the context, content or format of the data.

Parameter objects, on the other hand, provide a full description of all configurable data for a device. Since the device itself provides all the necessary information, a configuration tool can gain access to all parameters and maintain a user-friendly interface. However, this method imposes a burden on a device with full parameter information that may be excessive for a small device with limited internal resources. Therefore, an abbreviated version of the parameter object, called a parameter object stub, may be used (see Section 2.5.2). This option still allows access to the parameter data, but it does not describe any meaning to the data. parameter object stubs in conjunction with a printed data sheet are usable, but certainly not optimal. On the other hand, an EDS supplies all of the information that a full parameter object contains, in addition to I/O connection information, so the EDS provides the full functionality and ease of use of the parameter object without imposing an excessive burden on the individual device. In addition, an EDS provides a means for tools to perform offline configuration and to download configuration data to the device at a later time.

An EDS is a simple ASCII text file that can be generated on any ASCII editor. Since the CIP Specification provides a set of rules for the overall design and syntax of an EDS, specialized EDS editing tools, such as ODVA's EZ-EDS, can simplify the creation of EDS files. The main purpose of the EDS is to give information on several aspects of the device's capabilities, the most important ones being the I/O connections it supports and what parameters for display or configuration exist within the device. It is highly recommended that an EDS describe all supported I/O connections, as this makes the application of a device in a control system much easier. When it comes to parameters, EDS files should contain the attributes of application objects so that software can provide user access for monitoring and/or configuration purposes.

Let's look at some details of the EDS. First, an EDS is structured into sections, each of which starts with a section name in square brackets [ ]. The first two sections are mandatory for all EDS files.

- [File]: Describes the contents and revision of the file;

- [Device]: Is equivalent to the identity object information and is used to match an EDS to a device;

- [Device Classification]: Describes what network the device can be connected to. This section is optional for DeviceNet, required for ControlNet, EtherNet/IP and CompoNet;

- [ParamClass]: Describes configuration details in addition to class-level attributes of the parameter object;

- [Params]: Identifies all configuration parameters in the device, follows the parameter object definition. Further details below;

- [Groups]: Identifies all parameter groups in the device and lists group name and parameter numbers;
- [Assembly]: Describes the structure of data items;

- [Connection Manager]: Describes connections supported by the device. Typically used in ControlNet and EtherNet/IP;

- [Connection ManagerN]: Same as the [Connection Manager] section, but only for connection entries that do not apply to all CIP ports of the device;

- [Port]: Describes the various network ports a device may have;

- [Capacity]: Specifies the communication capacity of EtherNet/IP and ControlNet devices;

- [Connection Configuration]: This section defines the characteristics of the connection configuration object implemented in this device, if a connection configuration object implementation exists. It is used for EDS-based I/O Scanner configuration;

- [Event Enumeration]: The Event Enumeration section associates specific event or status codes within a device with an international string;

- [Symbolic Translation]: This section is used to publicize the translation between a Symbolic Segment or an ANSI Extended Symbol Segment encoded EPATH specification to the equivalent ParamN or AssemN entry keywords;

- [Internationalization]: This section allows the representation of all strings within an EDS in multiple languages;

- [Modular]: Describes modular structures inside a device;

- [IO_Info]: Describes I/O connection methods & I/O sizes. Allowed for DeviceNet only;

- [Variant_IO_Info]: Describes multiple IO_Info data sets. Allowed for DeviceNet only;

- [EnumPar]: Enumeration list of parameter choices to present to the user. This is an old enumeration method specified for DeviceNet only;

- [ControlNet Physical Layer]: Describes details of the ControlNet physical layer. Allowed for ControlNet only;

- [CompoNet_Device]: Describes the type of CompoNet device. Allowed for CompoNet only;

- [CompoNet_IO]: Describes the I/O connection details of CompoNet slaves. Allowed for CompoNet only;

- [Modbus Mapper]: Used to provide a description of individual Modbus items that correspond to a specific CIP object attribute.

- Object class sections: These sections – one for every object class – can be used to describe all object details, such as instances, attributes and supported services.

These sections allow a very detailed device description, although only a few of these details are described here. Further reading is available in endnotes [41], [42].

A tool with a collection of EDS files will first use the [Device] section to try to match an EDS with each device it finds on a network. Once this is done and a particular combination of device and EDS is chosen, the tool can then display device properties and parameters and allow their modification if the user so chooses. A tool may also display what I/O connections a device allows and which of these are already in use. EDS-based tools are mainly used for slave or I/O Adapter devices, as I/O Scanner devices typically are too complex to be configured through EDS constructs alone. For those devices, the EDS is used primarily to identify the device and then guide the tool to call a matching configuration applet.

A particular strength of the EDS approach lies in the methodology of parameter configuration. A configuration tool typically takes all of the information that can be supplied by the parameter objects and an EDS and displays it in a user-friendly manner. In many cases, this enables the user to configure a device without needing a detailed manual, as the tool presentation of the parameter information and the help text enables decision making for a complete device configuration. This assumes the developer of the product and the EDS file has supplied all required information and any optional information with completeness and accuracy.

A complete description of what can be done with EDS files goes well beyond the scope of this book. Available materials on this topic provide greater detail [41], [42].

2.8. CIP Routing

CIP defines mechanisms that allow the transmission of messages across multiple networks, provided that the intermediate devices (CIP routers) between the various networks are equipped with the objects and services used in CIP routing. If this is the case, the message will be forwarded from CIP router to CIP router until it has reached its destination node. Here is how it works:

For unconnected explicit messaging, the actual explicit message to be executed on the target device is "wrapped up" inside of another explicit message service, the so-called Unconnected_Send service (Service Code 0x52 of the connection manager object). This service message contains all the information about the transport mechanism, such as the request time-out (which may be modified as the message moves through each CIP router), the message request path information and the routing path information.

The first CIP router device that receives an Unconnected_Send message will take its contents and forward it to the next CIP router, as specified within the Route Path section of the message. Before the message is actually sent, the "used" part of the path is removed, but is remembered by the CIP router device for the return of the response. The CIP router may subtract some time from the timeout value, thereby reducing the timeout value as it closes in on the destination. This process is executed for every CIP router the message goes through, until the final CIP router is reached. The number of CIP routers an Unconnected_Send may pass through is theoretically limited by the message length.

Once the Unconnected_Send message has arrived at the last CIP router, the Unconnected_Send “wrapper” is removed and the “inner” explicit message is sent to the target device, which executes the requested service and generates a response. That response, as received from the target device, is then transported back through all the CIP routers it traversed during its forward journey until it reaches the originating node. It is important to note in this context that the transport mechanism may have been successful in forwarding the message and returning the response, but the response still could contain an indication that the desired service could not be performed successfully in the target network/device. Through this mechanism, the CIP router devices do not need to know anything about the message paths ahead of time so no pre-configuration of the CIP router devices is required. This is often referred to as “seamless routing”.

When a connection (I/O or Explicit) is set up using the Forward_Open service (see Section 3.1.14), it may go to a target device on another network. To enable the appropriate setup process, the Forward_Open message may contain a field with path information describing a route to the target device. This is very similar to the Unconnected_Send service described above. The routing information is then used to create routed connections within the CIP routing devices between the originator and the target of the message. Once set up, these connections automatically forward any incoming messages for this connection to the proper outgoing port. Again, this is repeated through each CIP router until the message has reached the target node. As with routed unconnected explicit messages, the number of hops is generally limited only by the capabilities of the devices involved. In contrast to routed unconnected messages, routed connected messages do not carry path information. Since connected messages always use the same path for any given connection, the path information that was given to the routing devices during connection setup is held there as long as the connection exists. Again, the CIP routing devices do not have to be preprogrammed; they are self-configured during the connection establishment process.

2.9. Data Management

The Data Management part of the CIP Specification describes addressing models for CIP entities and the data structures of the entities themselves.

Entity addressing is done by Segments, which allows flexible usage so that many different types of addressing methods can be accommodated. Two uses of this addressing scheme (logical segments and data types) are looked at in more detail below.

2.9.1. Logical Segments

Logical segments (first byte = 0x20 – 0x3F) are addressing Segments that can be used to address objects and their attributes within a device. They are typically structured as follows: [class ID] [instance ID] [attribute ID, if required].

Each element of this structure allows various formats (1 byte, 2 byte and 4 byte). Figure 10 shows a typical example of this addressing method.

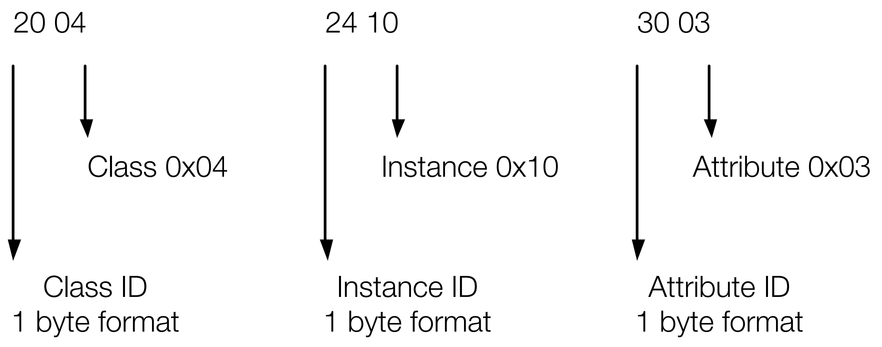


Figure 10 Logical Segment Encoding Example

This type of addressing is commonly used to point to assemblies, Parameters and other addressable entities within a device. It is used extensively in EDS files, but also within explicit messages, to name just a few application areas.

2.9.2. Data Types

Data types (first byte = 0xA0–0xDF) can be either structured (first byte = 0xA0–0xA3, 0xA8 or 0xB0) or elementary (first and only byte = 0xC1–0xDE). All other values are reserved. structured data types can be arrays of elementary data types or a collection of arrays or elementary data types. Of particular importance in the context of this book are elementary data types, which are used within EDS files to specify the data types of parameters and other entities.

Here is a list of commonly used data types:

- 1-bit (encoded into 1 byte):
  - Boolean, BOOL, Type Code 0xC1;

- 1-byte:
  - Bit string, 8 bits, BYTE, Type Code 0xD1;
  - Unsigned 8-bit integer, USINT, Type Code 0xC6;
  - Signed 8-bit integer, SINT, Type Code 0xC2;

- 2-byte:
  - Bit string, 16-bits, WORD, Type Code 0xD2;
  - Unsigned 16-bit integer, UINT, Type Code 0xC7;
  - Signed 16-bit integer, INT, Type Code 0xC3;

- 4-byte:
  - Bit string, 32 bits, DWORD, Type Code 0xD3;
  - Unsigned 32-bit integer, UDINT, Type Code 0xC8;
  - Signed 32-bit integer, DINT, Type Code 0xC4.

The data types in CIP follow the requirements of IEC 61131-3 [17] [16] .

2.10. Auxiliary Power Distribution System

The CIP application layer can be used on a variety of network technologies. Each CIP network specification consists of two volumes. The physical layer behavior defined on a particular network is described in the appropriate CIP network adaptation volume.

Chapter 8 of Volume 1 of the CIP Networks Library defines an optional auxiliary power distribution system that is separate and distinct from the physical layer requirements of any of the CIP networks.

Auxiliary power may be used to provide application power for such devices as input/output modules, Emergency Stop circuitry, and other application-specific needs. The cabling system provides 4-wire, two-circuit wiring that supplies 24V switched and un-switched power. Depending on the cabling selected by the designer, the maximum current ranges from 7 to 10 amperes. This standard specifies system topologies, cable and connector requirements and power supply requirements for auxiliary power distribution.

This system is not intended to provide redundant network power for already powered networks such as DeviceNet or CompoNet.

2.11. Maintenance and Further Development of the Specifications

ODVA has a set of working groups that maintain the specifications and create protocol extensions, e.g., new profiles or functional enhancements such as CIP Sync and CIP Safety. These groups are called Special Interest Groups (SIGs).

The results of these SIGs are written up and presented to the Technical Review Board (TRB) for approval and then incorporated into the specifications. Only ODVA members can work within the SIGs. These participants have the advantage of advance knowledge of technical changes coming to the specifications. Participation in one or several SIGs is, therefore, highly recommended.

3. Network Adaptations of CIP

There are currently four public adaptations of CIP, each based on different data link layers and transport mechanisms, which maintain the common upper layers of CIP, as illustrated earlier in Figure 1.

3.1. DeviceNet

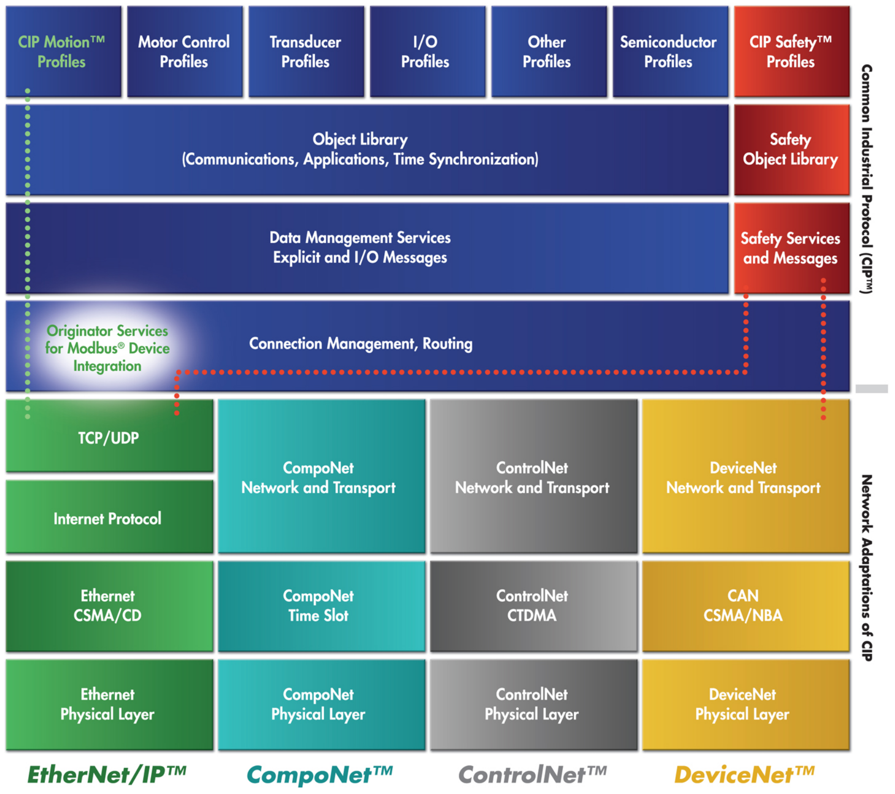
3.1.1. Introduction

DeviceNet was the first implementation of CIP. As mentioned in Section , DeviceNet is based on the Controller Area Network (CAN). DeviceNet uses a subset of the CAN protocol (11-bit identifier only, no remote frames). The DeviceNet adaptation of CIP accommodates the 8-byte packet size limitation of the CAN protocol and allows the use of simple devices with minimal processing power. For a more detailed description of the CAN protocol and some of its applications, see endnote [18].

3.1.2. Relationship to standards

Like other CIP Networks, DeviceNet follows the Open Systems Interconnection (OSI) model, an ISO standard for network communications that is hierarchical in nature. Networks that follow this model define all necessary functions, from physical implementation up to the protocol and methodology to communicate control and information data within and across networks.

Figure 11 shows the relationship between CIP and DeviceNet.



CIP Motion™ Profiles
Motor Control Profiles
Transducer Profiles
I/O Profiles
Other Profiles
Semiconductor Profiles
CIP Safety™ Profiles
Object Library (Communications, Applications, Time Synchronization)
Safety Object Library
Data Management Services Explicit and I/O Messages
Safety Services and Messages
Originator Services for Modbus® Device Integration
Connection Management, Routing
Common Industrial Protocol (CIP™)
TCP/UDP
Internet Protocol
CompoNet Network and Transport
ControlNet Network and Transport
DeviceNet Network and Transport
Ethernet CSMA/CD
CompoNet Time Slot
ControlNet CTDMA
CAN CSMA/NBA
Ethernet Physical Layer
CompoNet Physical Layer
ControlNet Physical Layer
DeviceNet Physical Layer
Network Adaptations of CIP
EtherNet/IP™
CompoNet™
ControlNet™
DeviceNet™

Figure 11 Relationship Between CIP and DeviceNet

The DeviceNet adaptation of CIP is described in Volume 3 of the CIP Networks Library [3]. All other features are based on CIP. DeviceNet is also described in a number of national and international standards, e.g. [25], [30].

3.1.3. DeviceNet Features

DeviceNet is a communication system at the low end (sensors, actuators) of the industrial communication spectrum with the following features:

- Trunkline/dropline configuration;
- Support for up to 64 nodes;
- Node insertion or removal while the network is up and running;
- QuickConnect for devices that are frequently removed from and added to the network, e.g., robot tools;
- Simultaneous support for both network-powered devices, e.g., sensors, and separately-powered devices, e.g., actuators;
- Use of sealed or open-style connectors;
- Protection from wiring errors;
- Selectable data rates of 125 kBaud, 250 kBaud and 500 kBaud;
- Adjustable power configuration to meet individual application needs;
- High current capability (up to 16 Amps per supply);
- Operation with off-the-shelf power supplies;
- Power taps that allow the connection of several power supplies from multiple vendors that comply with DeviceNet standards;
- Built-in overload protection;
- Power available along the bus: both signal and power lines contained in the cable;
- Several cables that are suitable for a number of different applications.

3.1.4. DeviceNet Physical Layer and Relationship to CAN
The physical layer of DeviceNet is an extension of the ISO 11898 standard [19]. This extension defines the following additional details:

- Improved transceiver characteristics that allow the support of up to 64 nodes per network;

- Additional circuitry for over-voltage and mis-wiring protection;

- Several types of cables for a variety of applications;

- Several types of connectors for open (IP 20) and sealed (IP 65/67) devices.

The cables described in the CIP Networks Library were designed specifically to meet minimum propagation speed requirements to ensure that they can be used up to the maximum system length. Figure 12 shows examples of some of the key characteristics that can be achieved with some of the defined cable types in conjunction with suitable transceiver circuits and proper termination resistors (121 Ω).

	Trunk Distance			Drop Length	
Data Rate	Thick Cable	Thin Cable	Flat Cable	Maximum	Cumulative
125 kBaud	500 meters	100 meters	420 meters	6 meters	156 meter
250 kBaud	250 meters		200 meters		78 meters
500 kBaud	100 meters		75 meters		39 meters

Figure 12 Data Rate vs. Trunk and Drop Length

ODVA has issued a guideline [8] that gives complete details of how to build the physical layer of a DeviceNet Network, equivalent information can also be found in an IEC standard [27].

Developers of DeviceNet devices can create DeviceNet circuits with or without physical layer isolation (both versions are fully specified). Furthermore, a device may take some or all of its power from the bus, thus avoiding extra power lines for devices that can live on the power supplied through the DeviceNet cable.

All DeviceNet devices must be equipped with one of the connectors described in Volume 3, although hard wiring of a device is allowed, provided the node is removable without severing the trunk.

3.1.5. Frame Structure
DeviceNet uses standard CAN frames with an 11-bit identifier, for further details see [19], [20] and chapter 3.1.15 of this publication.

3.1.6. Protocol Adaptations

On the protocol side, there are basically two adaptations of CIP that have been made to better accommodate it to the CAN data frame:

- Shortening CIP explicit messages to 8 bytes or less where possible, with the use of message fragmentation for longer messages;

- Definition of a master/slave communications option to minimize the connection establishment overhead (see chapter 3.1.17).

These two features have been created to allow the use of simple and thus inexpensive microcontrollers. This is particularly important for small, cost-sensitive devices like photoelectric or proximity sensors. As a result of this adaptation, the DeviceNet protocol in its simplest form has been implemented in 8-bit microprocessors with as little as 4 Kbytes of code memory and 175 bytes of RAM.

The message fragmentation mentioned previously comes in two varieties:

- For I/O messages the use of fragmentation is defined by the maximum length of the data to be transmitted through a connection. Any connection that has more than 8 bytes to transmit always uses the fragmentation protocol, even if the actual data to be transmitted is 8 bytes or less, e.g., an “Idle” Message.

- For explicit messaging, the use of the fragmentation protocol is indicated in the header of every message, since the actual frame size can vary in length, depending on the content of the explicit message. The actual fragmentation protocol is contained in one extra byte within the message that indicates whether the fragment is a start fragment, a middle fragment or an end fragment. A modulo 64 rolling fragment counter allows very long fragmented messages, and is limited in theory only by the maximum Produced or Consumed Connection sizes (65,535 bytes). In reality, the capabilities of the devices limit the message sizes.

3.1.7. Indicators and Switches

Indicators and switches are optional on DeviceNet. However, certain DeviceNet users not only require indicators and switches, they also specify what type to use. Many factors must be considered before implementing these devices, including packaging, accessibility and customer expectations.

Indicators allow the user to determine the state of the device and its network connection(s). Since indicators can be very useful when troubleshooting the operation of a device, manufacturers are advised to incorporate some or all of the indicators described in the DeviceNet specification. While devices may incorporate additional indicators with behavior not described in the specification, any indicators labeled per specification must also follow their specified behavior.

Similarly, devices may be built with or without switches or other directly accessible means for configuration of MAC ID and baud rate. If these switches are used, certain rules apply to how these values are used at power-up and during the operation of the device.

3.1.8. Additional Objects

The DeviceNet Specification defines one additional object, the DeviceNet object.

3.1.8.1. DeviceNet Object (Class ID: 0x03)

A DeviceNet object is required for every DeviceNet port of the device. The instance attributes of this object contain information on how this device uses the DeviceNet port, including the MAC ID of the device and the (expected) baud rate of the DeviceNet network the device is attached to. Both attributes are always expected to be non-volatile, i.e., after a power interruption, the device is expected to try to go online again with the same values that were stored in these attributes before the power interruption. Devices that set these values through switches typically override any stored values at power-up. The DeviceNet object may also contain information on further aspects associated with its DeviceNet behavior, such as information related to the master/slave communications status, QuickConnect support (see chapter 3.1.17.7) and Active Node Table.

3.1.9. Network Access

DeviceNet uses the network access mechanisms described in the CAN specification, i.e., bitwise arbitration through the CAN Identifier for every frame to be sent. This requires a system design that does not allow multiple uses of any of these identifiers. Since the node number of every device is coded into the CAN Identifier (see chapter 3.1.15), it is generally sufficient to make sure that none of the node numbers exists more than once on any given network. This is guaranteed through the Network Access algorithm (see chapter 3.1.10).

3.1.10. Going Online

Any device that wants to communicate on DeviceNet must go through a network access algorithm before any communication is allowed. The main purpose of this process is to avoid duplicate node IDs on the same network, a secondary purpose is to announce a node's presence on the link for nodes that maintain an Active Node Table. Every device that is ready to go online sends a Duplicate MAC ID Check Message containing its port number, vendor ID and serial number. If another device is already online with this MAC ID or is in the process of going online with this MAC ID, it responds with a Duplicate MAC ID Response Message that directs the checking device to go offline and not communicate any further.

If two or more devices with the same MAC ID happen to transmit the Duplicate MAC ID Check Message at exactly the same time, all of them will win arbitration at the same time and will proceed with their message. However, since this message has different values (port number, vendor ID and serial number) in the data field, the nodes will detect bit errors and will flag error frames that cause all nodes to discard the frame. This reaction triggers a re-transmission of the message by the sending node. While this action may eventually result in a Bus-Off condition for the devices involved, a situation with duplicate node IDs is safely avoided.

3.1.11. Offline Connection Set

The Offline Connection Set is a set of messages created to communicate with devices that have failed to go online (see Section 3.1.10), to allow a new MAC ID to be set. At any given point in time, only one offline device and one tool can use the Offline Connection Set, therefore, the first step in its use is to determine if a tool has ownership of the Offline Connection Set. Once a tool has successfully claimed ownership, it can check whether there are any nodes on the network that are in the offline state. If such nodes exist, the tool can then determine their vendor ID(s) and serial number(s). Using this information, which is unique by definition, the tool can then address a specific device which responds by flashing an indicator. Once this identification is complete and the user is certain communication is established with the intended device, the tool can then send a new MAC address to the device. The target device then restarts the Duplicate MAC ID algorithm and tries to go online with the new MAC Address. More information on this topic can be found in [3] and [18].

3.1.12. DeviceNet Status Indication Messages

There are two optional DeviceNet messages that indicate a status or a status transition of a device. One of them is called “Device Heartbeat” and the other is called “Device Shutdown”. Both messages are transmitted by a UCMM capable device as an unconnected response message (Message Group 3, Message ID 5) and by a Group 2 Only Server as an unconnected response message (Message Group 2, Message ID 3). These messages are independent of any other communication relationship that may exist with other devices on the network.

The Device Heartbeat Message, sent at a heartbeat interval set in the ID object, provides a way for a device to indicate its presence on the network and its current “health” condition. The Device Shutdown Message provides a way for a device to indicate that it is in the process of shutting down and going to the offline state.

3.1.13. Explicit Messaging

All explicit messaging in DeviceNet is done via connections and the associated connection object instances. However, these objects first must be set up in the device. This can be done by using the predefined master/slave connection set to activate a static connection object already available in the device or by using the UCMM (unconnected message manager) port of a device, to dynamically set up a connection object for explicit messaging. The only messages sent to the UCMM are “Open” or “Close” requests that set up or tear down an explicit messaging connection, while the only messages that can be sent to the master/slave equivalent of the UCMM called the Group 2 Only Unconnected Port, are the “Allocate” or “Release” service requests (see chapter 3.1.17). Explicit messages always pass via the message router object to the object that is being addressed (refer to Figure 8).

As mentioned in Section 2.3, explicit messages on DeviceNet have a very compact structure to make them fit into the 8-byte frame in most cases. Figure 13 shows a typical example of a request message using the 8/8 Message Body Format (“8/8” means 1 byte for Class ID, 1 byte for Instance ID).

	Bit number								
Byte offset	7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID						Message header
1	R/R [0]	Service Code							Message body
2	Class ID								
3	Instance ID								
4 …	Service data …								

Figure 13 Non-Fragmented Explicit Request Message Format

The consumer of this explicit message responds using the format shown in Figure 14. The consumer sets the R/R (Request/Response) bit and repeats the Service Code of the request message. Any data transferred with the response is entered in the service data field.

	Bit number								
Byte offset	7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID						Message header
1	R/R [1]	Service Code							Message body
2 … 7	Service data … (optional)								

Figure 14 Non-Fragmented Explicit Response Message Format

Most messages will use the 8/8 format shown in Figure 13, since they only need to address Class and Instance IDs up to 255. If there is a need to address any class/instance combinations above 255, then this is negotiated between the two communication partners during the setup of the connection. Should an error occur, the receiver responds with the Error Response Message. The Service Code for an Error Response message is 0x14, and two bytes of error code are included in the service data field to convey more information about the nature of the error. See endnotes [3], [18] for further details of message encoding, including the use of fragmentation.

3.1.14. I/O Messaging

Since DeviceNet does not use a Real-Time Header or sequence count value like ControlNet and EtherNet/IP do, I/O messages in DeviceNet have a very compact structure. For I/O data transfers up to 8 bytes long, the data is sent in a non-fragmented message, which uses the entire CAN data field for I/O data. For I/O data transfers longer than 8 bytes, a fragmentation protocol spreads the data over multiple frames. This fragmentation protocol uses one byte of the CAN data field to control the fragmentation of the data. See Figure 15 and Figure 16 for examples of fragmented and non-fragmented I/O messages. I/O messages without data (i.e., with zero length data) indicate the "Idle" state of the producing application. Any producing device can do this – master, slave or peer.

	Bit number							
Byte offset	7	6	5	4	3	2	1	0
0 … 7	Process data (0 – 8 bytes)							

Figure 15 Non-Fragmented I/O Message Format

	Bit number							
Byte offset	7	6	5	4	3	2	1	0
0	Fragmentation protocol							
1 … 7	Process data (0 – 7 bytes)							

Figure 16 Fragmented I/O Message Format

As mentioned, I/O messages are used to exchange high-priority application and process data via the network, and this communication is based on the producer/consumer model. The associated I/O data are always transferred from one producing application object to one or more consuming application objects. This is accomplished using I/O messages via I/O messaging connection objects (Figure 17 shows two consuming applications) that have been pre-set in the device. This can be done in one of two ways by using:

- The predefined master/slave connection set to activate a static I/O connection object already available in the device;
- An explicit messaging connection object already available in the device to dynamically create and set up an appropriate I/O connection object.

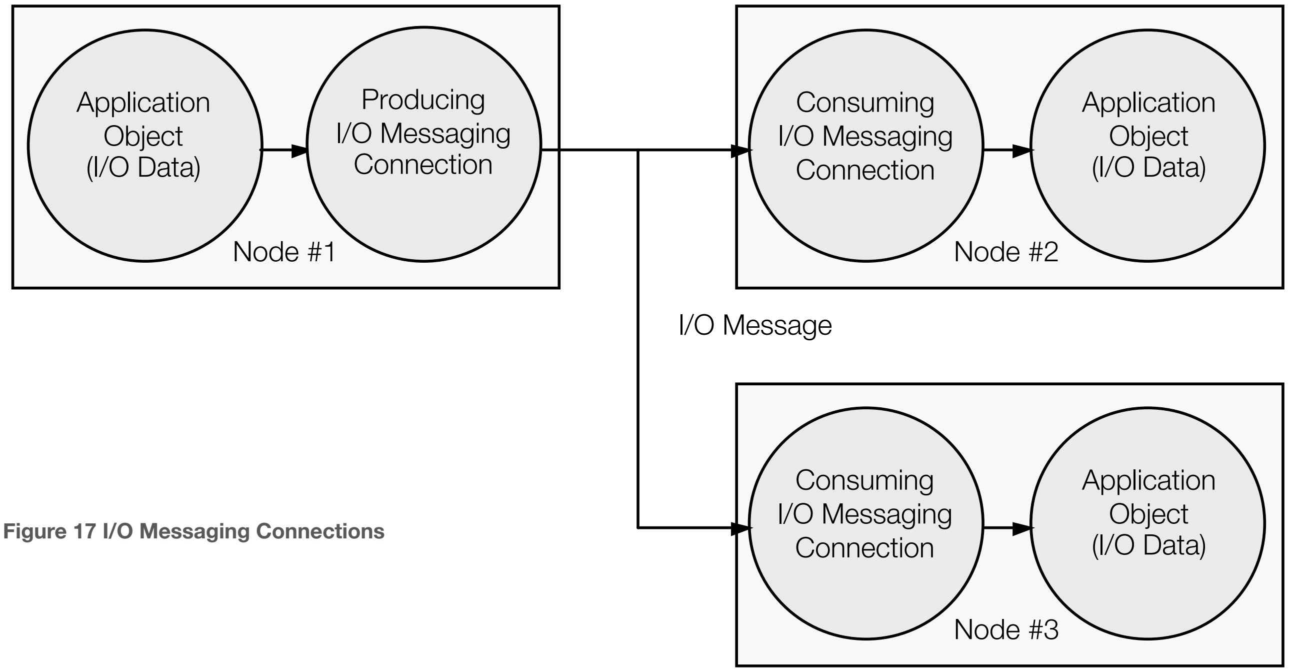

Application Object (I/O Data)
Producing I/O Messaging Connection
Node #1
Consuming I/O Messaging Connection
Application Object (I/O Data)
Node #2
I/O Message
Consuming I/O Messaging Connection
Application Object (I/O Data)
Node #3

Figure 17 I/O Messaging Connections

I/O messages usually pass directly to the data of the assigned application object. The assembly object is the most common application object used with I/O connections. Also refer to Figure 8.

3.1.15. Using the CAN Identifier

DeviceNet is based on the standard CAN protocol and therefore uses an 11-bit message identifier. A distinction therefore can be made between 211 = 2048 messages. The 6-bit MAC ID field is sufficient to identify a device because a DeviceNet Network is limited to a maximum of 64 participants.

The overall CAN Identifier range is divided into four Message Groups of varying sizes, as shown in Figure 18:

Connection ID = CAN Identifier (bits 10:0)											Used for
10	9	8	7	6	5	4	3	2	1	0	
0	Message ID				Source MAC ID						Message Group 1
1	0	MAC ID						Message ID			Message Group 2
1	1	Message ID			Source MAC ID						Message Group 3
1	1	1	1	1	Message ID						Message Group 4
1	1	1	1	1	1	1	x	x	x	x	Invalid CAN Identifiers

Figure 18 Definition of the Message Groups

The bitwise arbitration mechanism of CAN determines the priority of messages on DeviceNet. When two nodes transmit simultaneously, the numerically lower CAN Identifier value will win arbitration. The arbitration mechanism is explained in the CAN specification [20]. A detailed description is beyond the scope of this document, but in short, transmitted bits are shifted onto the wire most significant bit first, so a zero in the upper bit positions will take precedence over a one. As is shown in Figure 18, Message Group 1 has a zero in bit 10, so it is the highest priority group. Group 2 is the second highest priority group because of the zero in bit 9. Group 3 is the next highest priority group, because the CAN IDs contain a one in bits 9 and 10. All valid Group 3 Message IDs are lower numerically than the corresponding bits (8-6) in Group 4 and therefore Group 4 is the lowest priority of all. In DeviceNet, the CAN Identifier is the connection ID. This comprises the Message Group ID, the Message ID within this group and the device's MAC ID, which can be the source or destination address. The definition depends on the Message Group and the Message ID. The significance of the message within the system is defined by the Message Group and Message ID.

The four Message Groups are used as follows:

Message Group 1 is assigned 1024 CAN Identifiers (0x0000 – 0x03FF), which is 50 % of all identifiers available. Up to 16 different Message IDs are available per device (node) within this group. The priority of a message within this group is primarily determined by the Message ID (the significance of the message), and only after that by the source MAC ID (the producing device). If two devices transmit a Message Group 1 message at the same time, then the device with the lower Message ID will always win the arbitration. However, if two devices transmit the same Message ID at the same time on the CAN bus, then the device with the lower MAC ID will win. The messages of Group 1 are, therefore, well suited for the exchange of high-priority process data.

Message Group 2 is assigned 512 identifiers (0x0400 – 0x05FF). Most of the Message IDs in this group are optionally defined for what is commonly referred to as the predefined master/slave connection set (see chapter 3.1.17). One Message ID is reserved for the Duplicate Node ID Check (see chapter 3.1.10). Priority within Message Group 2 is determined primarily by the MAC ID and, only after that, by the Message ID. This message group was designed so that a CAN controller with an 8-bit mask is able to filter out its Group 2 Messages based on MAC ID. This makes it possible for very low cost, low functionality microcontrollers with integral CAN controllers to be suitable for use on DeviceNet.

Message Group 3, with 448 CAN Identifiers (0x0600 – 0x07BF), has a similar structure to Message Group 1, however, it is mainly used for low priority process data exchange due to the relative priority difference between Groups 1 and 3. In addition, the main use of this group is setting up dynamic Explicit Connections. Seven Message IDs per device are possible, and two of these are reserved for what is commonly referred to as the UCMM port (see chapter 3.1.16).

Message Group 4, with 48 CAN Identifiers (0x07C0 – 0x07EF), does not include any MAC IDs, only Message IDs. The messages in this group are only used for network management. Four Message IDs are currently assigned for services of the Offline Connection Set.

The remaining 16 CAN Identifiers (0x07F0 – 0x07FF) are invalid CAN IDs and thus are not permitted for use in DeviceNet systems.

With this allocation of CAN Identifiers, the unused CAN Identifiers cannot be used by other devices. Therefore, each device has exactly 16 Message IDs in Group 1, eight Message IDs in Group 2 and seven Message IDs in Group 3. One advantage of this system is that the CAN Identifiers used in the network can always be clearly assigned to a device. Devices are responsible for managing their own identifiers. This simplifies the design, troubleshooting and diagnosis of DeviceNet systems, as a central tool that keeps a record of all CAN ID assignments on the network is not needed.

3.1.16. Connection Establishment

As described in chapters 3.1.12 and 3.1.14, messages on DeviceNet are always exchanged in a connection-based manner. Communication objects must be set up for this purpose. These are not initially available when a device is powered on; they first have to be created. There are two ports by which a DeviceNet device can be addressed when first powered on, the unconnected message manager port (UCMM port) or the Group 2 Only Unconnected Explicit Request port, which is defined by the predefined master/slave connection set. Picture these ports as doors to the device. Only one key will unlock each door. The appropriate key for each lock is the connection ID – i.e., the CAN Identifier – of the selected port. Other doors in the device can be opened only if and when the appropriate key is available and other instances of connection objects are set up.

The setting up of communication relationships (i.e., connections) via the UCMM port represents a general procedure that should be adhered to with all DeviceNet devices. Devices that feature the predefined master/slave connection set and are UCMM capable are called Group 2 Servers. A Group 2 Server can be addressed by one or more connections from one or more clients.

Since UCMM capable devices need a good amount of processing power to service multiple communication requests, a simplified communication establishment and I/O data exchange method has been created for low-end devices. This is called the predefined master/slave connection set (see chapter 3.1.17). This covers as many as five predefined connections that can be activated (assigned) when accessing the device. The predefined master/slave connection set represents a subset of the general connection establishment method, and it is limited to pure master/slave relations. Slave devices that are not UCMM capable and only support this subset are called Group 2 Only Servers. Only the master that allocates it can address a Group 2 Only Server. All messages received by this device are defined in Message Group 2.

For more details of the connection establishment using UCMM and the master/slave connection set, refer to endnotes [3], [18].

3.1.17. Predefined Master/Slave Connection Set

Establishing a connection via the UCMM port requires a relatively large number of steps that must be completed to allow data exchange via DeviceNet and the devices must provide resources to administer the dynamic connections. Because every device can set up a connection with every other device, and the source MAC ID of the devices is contained in the connection ID, the CAN Identifier (connection ID) may have to be filtered via software. This depends on how many connections a device supports, and on the type and number of screeners (hardware CAN ID filters) of the CAN chip used in the device's implementation.

While this approach maximizes use of the multicast, peer-to-peer, and producer/consumer capabilities of CAN, it requires a higher performance CPU and more RAM and ROM resources. These requirements eliminate an entire class of low cost microcontrollers with internal CAN controllers from consideration, raising the cost of implementation to levels that preclude cost effective solutions for low-end (e.g., low end-user cost) devices. The predefined master/slave connection set was designed to minimize message processing and to take advantage of the limited screening capabilities of many CAN controllers. The predefined master/slave connection set is the way that the vast majority of devices communicate on DeviceNet.

The predefined master/slave connection set defines an alternate way to establish connections called the Group 2 Only Unconnected Explicit Request Port. This method allows a device to limit the messages received to only those in Group 2 with its own MAC ID. This greatly reduces the amount of packets that a node must deal with. A CAN controller with a single mask and match screener (a so called BasicCAN screener) can be used in this type of device, which makes it possible to use the low cost microcontrollers and simple CAN controllers mentioned earlier. Devices that operate in this manner are referred to as Group 2 Only Servers, deriving their name from the fact that they are only required to receive messages in Group 2.

The predefined master/slave connection set is also used in UCMM capable devices. Such devices are referred to as Group 2 Servers, deriving their name from the fact that they respond to Group 2 messages but are not limited to just Group 2 messages.

Many of the reasons for defining the predefined master/slave connection set were due to hardware limitations prevalent when the protocol was first developed. Many of the cost considerations have changed as hardware evolved over time. Today, most devices can be implemented with a UCMM port and still be cost effective. This is the preferred type of device to develop today. For reasons that go beyond this scope of this document, devices that are not capable of UCMM place extra burden on other devices and tools. Except in extremely low cost situations, UCMM should always be a design goal for DeviceNet products.

The predefined master/slave connection set provides an interface for a set of up to five pre-configured connection types in a node.

The basis of this model is a 1:n communication structure consisting of one control device and decentralized I/O devices. The central portion of such a system is known as the "Master" and the decentralized devices are known as "Slaves". Multiple masters are allowed on the network, but a slave can only be allocated to one master at any time.

The predefined connection objects occupy instances 1 to 5 in the connection object (Class ID 0x05, see Section 2.4):

- Explicit Messaging Connection:

  - Group 2 Explicit Request/Response Message (Instance ID 1);

- I/O Messaging Connections:

  - Polled I/O connection (Instance ID 2);

  - Bit-strobe I/O connection (Instance ID 3);

  - Change of State or cyclic I/O connection (Instance ID 4);

  - Multicast Polling I/O connection (Instance ID 5).

The messages to the slave are defined in Message Group 2, and some of the responses from the slave are contained in Message Group 1. The distribution of connection IDs for the predefined master/slave connection set is defined as shown in Figure 19.

Connection ID = CAN Identifier (bits 10:0)											Used for
10	9	8	7	6	5	4	3	2	1	0	
0	Group 1 Message ID				Source MAC ID						Group 1 Messages
0	1	1	0	0	Source MAC ID						Slave's I/O Multicast Poll Response
0	1	1	0	1	Source MAC ID						Slave's I/O Change of State or Cyclic Message
0	1	1	1	0	Source MAC ID						Slave's I/O Bit-Strobe Response Message
0	1	1	1	1	Source MAC ID						Slave's I/O Poll Response or COS/Cyclic Ack Message
1	0	MAC ID						Group 2 Message ID			Group 2 Messages
1	0	Source MAC ID						0	0	0	Master's I/O Bit-Strobe Command Message
1	0	Source MAC ID						0	0	1	Master's I/O Multicast Poll Group ID
1	0	Destination MAC ID						0	1	0	Master's Change of State or Cyclic Ack Message
1	0	Source MAC ID						0	1	1	Slave's Explicit/Unconnected Response Messages
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages
1	0	Destination MAC ID						1	0	1	Master's I/O Poll Command/COS/Cyclic Message

Figure 19 Connection IDs of the Predefined Master/Slave Connection Set

Because the CAN ID of most of the messages the master produces contains the destination MAC ID of the slave, it is imperative that only one master talks to any given slave. Therefore, before it can use this Predefined Connection Set, the master must first allocate it with the device. The DeviceNet object manages this important function in the slave device. It allows only one master to allocate its Predefined Connection Set, thereby preventing duplicate CAN IDs from appearing on the wire.

The two services used are called Allocate_Master/Slave_Connection_Set (Service Code 0x4B) and Release_ Group_2_Identifier_Set (Service Code 0x4C). These two services always access instance 1 of the DeviceNet object (Class ID 0x03) (see Figure 20).

Byte offset	Bit number 7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID						Message header
1	R/R [0]	Service Code [0x4B]							Message body
2…5	Class ID [0x03]								
	Instance ID [0x01]								
	Allocation Choice								

Figure 20 Allocate_Master/Slave_Connect_Set request message

Figure 20 shows the Allocate Message with 8-bit Class ID and 8-bit Instance ID, a format that is always used when it is sent as a Group 2 Only unconnected message. It also may be sent across an existing connection and in a different format if a format other than 8/8 was agreed during the connection establishment.

The Allocation Choice Byte is used to determine which predefined connections are to be allocated (see Figure 21).

Bit number							
7	6	5	4	3	2	1	0
Reserved	Ack Suppression	Cyclic	Change of State	Multicast Polling	Bit-Strobe	Polled	Explicit Message

Figure 21 Format of the Allocation Choice Byte

The associated connections are activated by setting the appropriate bits. Change of State and Cyclic are mutually exclusive choices. The Change of State/Cyclic Connection may be configured as not acknowledged using the acknowledge suppression bit. The individual connection types are described in more detail below.

The allocator's MAC ID contains the address of the node (master) that wants to assign the predefined master/slave connection set. Byte 0 of this message differs from the allocator's MAC ID if this service has been passed on to a Group 2 Only Server via a Group 2 Only Client (what is commonly referred to as a "proxy function").

The slave, if not already claimed, responds with a Success Message. The connections are now in the Configuring State. Setting the Expected_Packet_Rate EPR (Set_Attribute_Single service to attribute 9 in the appropriate connection object instance, value in ms) starts the connection's time-monitoring function. The connection then changes into Established State, and I/O messages begin transferring via this connection.

The allocated connections can be released individually or collectively through the Release_ Master/Slave_Connection_Set service (Service Code 0x4C), using the same format as that in Figure 20, except that the last byte (Allocator's MAC ID) is omitted.

The following is an explanation of the four I/O connection types in the predefined master/slave connection set.

3.1.17.1. Polled I/O Connection

A polled I/O connection is used to implement a classic master/slave relationship between a control unit and a device. In this setup, a master can transfer data to a slave using the Poll Request and receive data from the slave using the Poll Response. Figure 22 shows the exchange of data between one master and three slaves in Polled I/O mode.

The amount of data transferred in a message between a master and a slave using the polled I/O connection can be any length. If the length exceeds 8 bytes, the fragmentation protocol is automatically used. A polled I/O connection is always a point-to-point connection between a master and a slave. The slave consumes the Poll Message and sends back an appropriate response (normally, its input data).

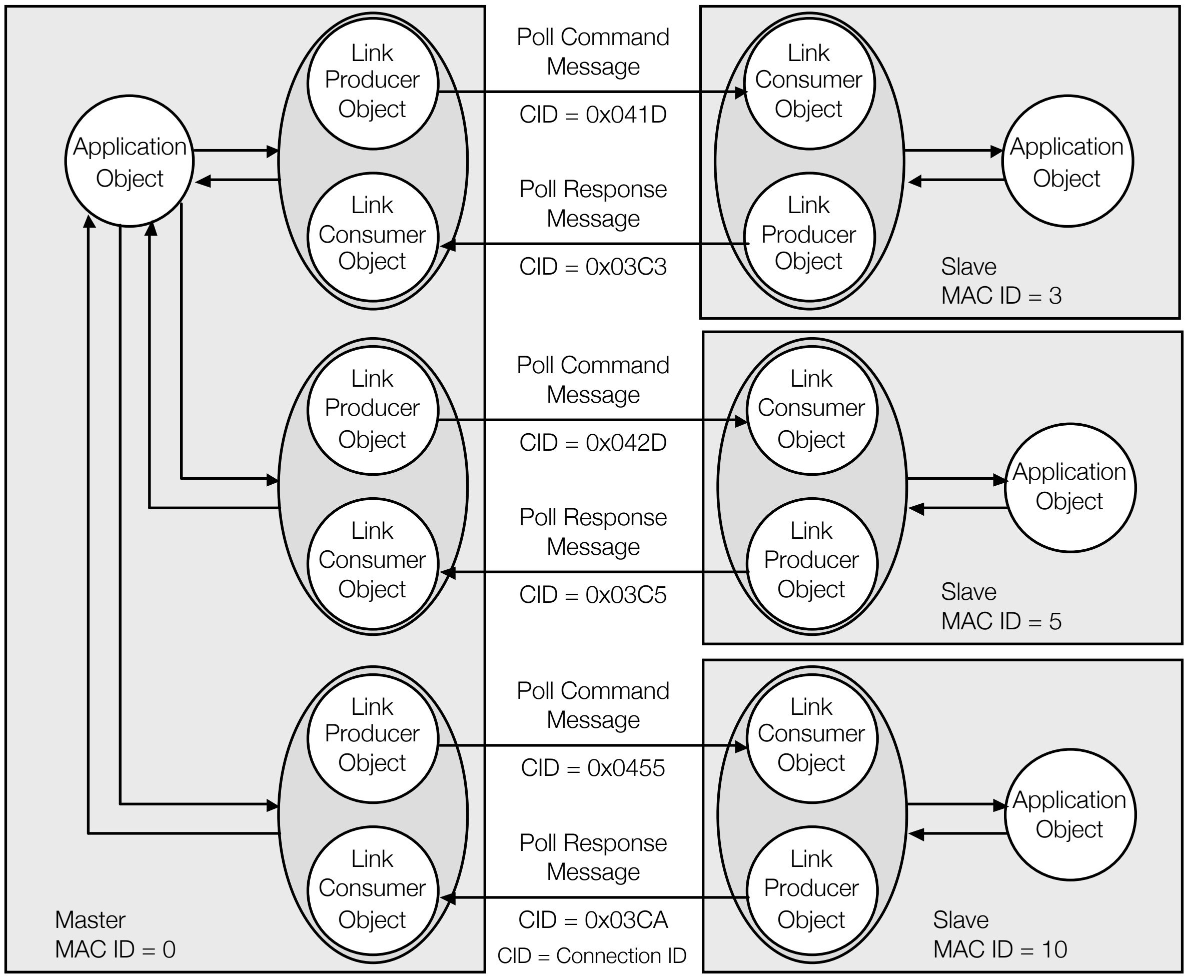


Figure 22 Polled I/O Connections

The Polled Connection is subject to a time-monitoring function, which can be adjusted, in the device. A Poll Command must have been received within this time (4 × EPR) or else the connection reverts to time-out mode. When a connection times out, the node optionally may go to a preconfigured fault state as set up by the user. A master usually polls all the slaves in a round-robin manner.

A slave's response time to a poll command is not defined in The DeviceNet Specification. While this provides flexibility for slave devices to be tailored to their primary application, it may also exclude the device from use in higher-speed applications.

3.1.17.2. Bit-Strobe I/O Connection

The master's transmission on this I/O connection is the bit-strobe command. Using this command, a master multicasts one message to reach all its slaves allocated for the bit-strobe connection. The frame sent by the master using a bit-strobe command is always 8 bytes or 0 bytes (if Idle). From these 8 bytes, each slave is assigned one bit (see Figure 23). Each slave can send back as many as 8 data bytes in its response.

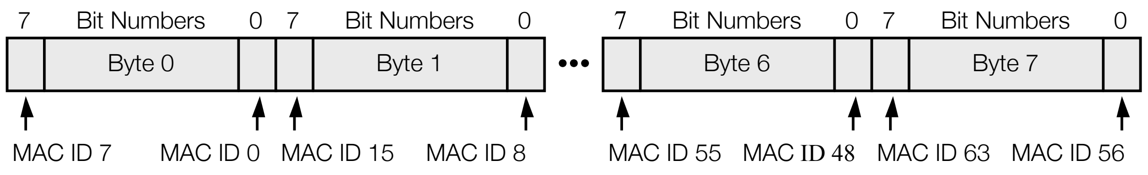


Figure 23 Data Format of the Bit-Strobe I/O Connection

A bit-strobe I/O connection represents a multicast connection between one master and any number of strobe-allocated slaves (see Figure 24). Since all devices in a network receive the bit-strobe command at the same time, they can be synchronized by this command. When the bit-strobe command is received, the slave may consume its associated bit, and then send a response of up to 8 bytes.

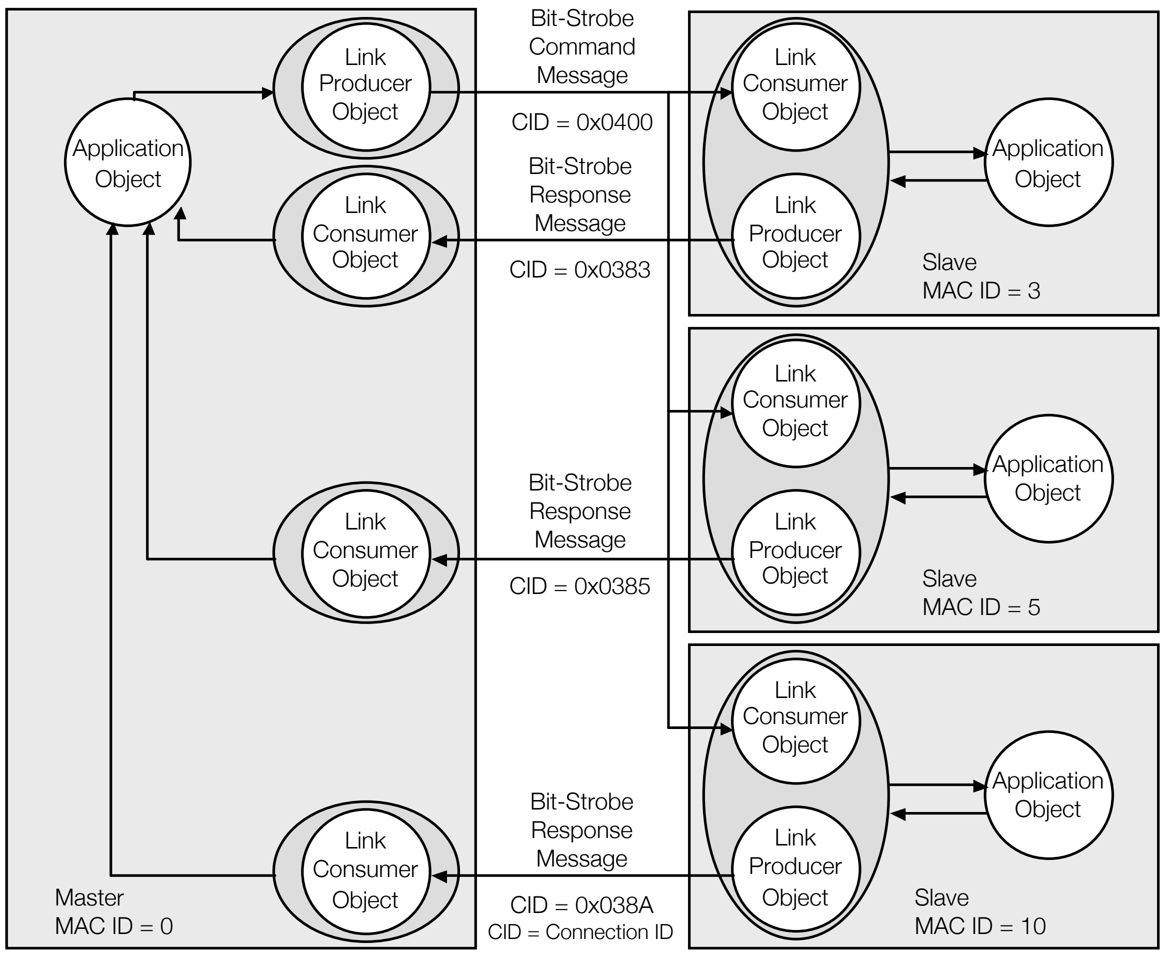


Figure 24 Bit-Strobe I/O Connections

Since this command uses the source MAC ID in the connection ID (see Figure 19), devices that support the bit-strobe I/O connection and have a CAN chip with screening limited to only 8 bits of the CAN ID (11 bits) must perform software screening of the CAN Identifier.

3.1.17.3. Change of State/Cyclic I/O Connection

The COS/cyclic I/O connection differs from the other types of I/O connections in that both endpoints produce their data independently. This can be done in a change of state or cyclic manner. In the former case, the COS I/O connection recognizes that the application object data indicated by the Produced_Connection_Path has changed. In the latter case, a timer of the cyclic I/O connection expires and therefore triggers the message transfer of the latest data from the application object.

A COS/cyclic I/O connection can be set up as acknowledged or unacknowledged. When acknowledged, the consuming side of the connection must define a path to the acknowledge handler object to ensure proper handling of acknowledgments and management of any required retries.

Figure 25 shows the various COS/Cyclic I/O connection possibilities.

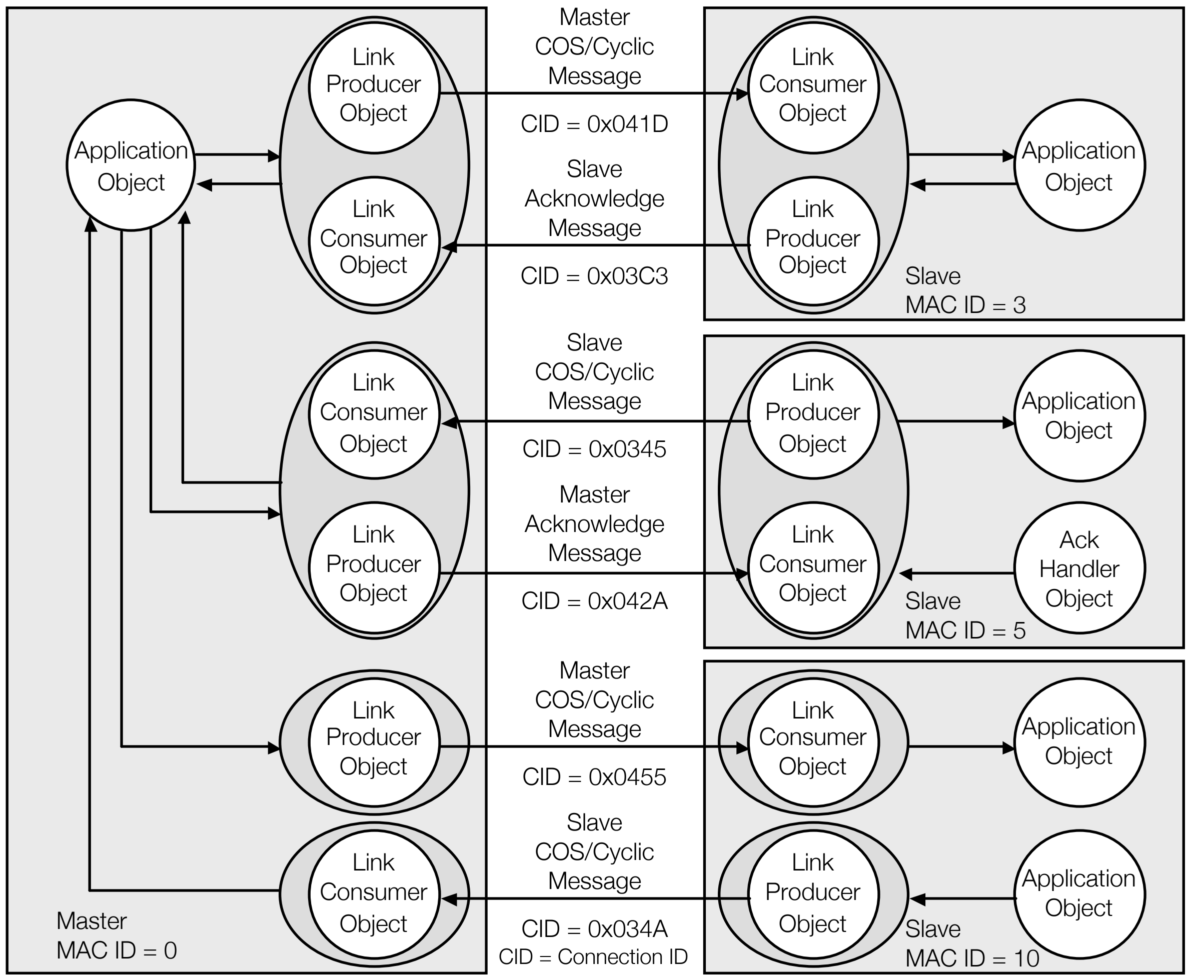


Figure 25 COS/Cyclic I/O Connections

A COS/cyclic I/O connection can also originate from a master, making it appear to the slave like a polled I/O connection. This can be seen in Figure 19 since the same connection ID is used for the master's polled I/O message as is used for the master's COS/cyclic I/O message. COS Connections have two additional behaviors not present in other connection types. The Expected Packet Rate (EPR) is used as a default production trigger so that, if the connection data have not changed after the EPR timer has expired, it will be resent anyway. This "heartbeat", as it is sometimes called, is utilized so the consuming node can know the difference between a node that has gone offline and one whose data have not changed. COS Connections also have a Production Inhibit Timer feature that prevents a node from producing data too often, and thus using too much network bandwidth. The production inhibit timer determines the amount of time the node must remain quiet after producing data to the network.

3.1.17.4. Multicast Polled I/O Connection

This connection is similar to the regular I/O poll except that all of the slaves belonging to a multicast group consume the same output data from the master. Each slave responds with its own reply data. A unique aspect of this connection is that the master picks the CAN ID from one of the slaves in the multicast group and must then set the consumed CAN ID in each of the other slaves to that same value. If, during runtime, that slave's connection times out, the master must either stop producing its multicast poll command or pick another slave in the group and reset the command CAN ID in all the remaining slaves in the group to that value before sending another Multicast Poll Command.

3.1.17.5. I/O Data Sharing

Due to the inherent broadcast nature of all CAN frames, applications can be set up to "listen" to the data produced by other applications. Such a "listen only" mode is not described in the DeviceNet specification, but some vendors have created products that do exactly that, e.g., "shared inputs" in Allen-Bradley I/O Scanners.

3.1.17.6. Typical Master/Slave Start Sequence

Typically, starting up a DeviceNet Network with an I/O Scanner and a set of slaves is executed as follows:

- All devices run their self-test sequence and then try to go online with the algorithm described in Section 3.1.10. Any device that uses an autobaud mechanism to detect the baud rate of a network has to wait with its Duplicate Node ID Message until it has seen enough CAN frames to detect the correct baud rate;

- Once online, slave devices will remain silent, except to defend their MAC ID, until their master allocates them;

- Once online, a master will try to allocate each slave configured into its scan list by running the following sequence of messages:

  - Try to open a connection to the slave using a UCMM Open Message;

  - If successful, the master can then use this connection for further communication with the slave (the device is a Group 2 Server);

  - If not successful, the master will try again after a minimum wait time of one second;

  - If unsuccessful again, the master will try to allocate the slave using the Group 2 Only Unconnected Explicit Request Message (at least for explicit messaging) after a minimum wait time of one second;

  - If successful, the master can then use this connection for further communication with the slave (the device is a Group 2 Only Server);

  - If not successful, the master will try again after a minimum wait time of one second;

  - If unsuccessful again, the master will start over with the UCMM Message after a minimum wait time of one second. This process will carry on indefinitely or until the master has allocated the slave.

- Once the master has allocated the slave, it may carry out some identity verification to see whether it is safe to start I/O messaging with the slave. The master also may apply further configuration to the connections it has established, e.g., setting the explicit messaging connection to "Deferred Delete";

- Setting the EPR value(s) brings the I/O connection(s) to the Established State so that I/O messaging can commence.

3.1.17.7. QuickConnect Connection Establishment

DeviceNet also allows an optional method of connection establishment known as QuickConnect. This was designed to provide the same level of protection against duplicate MAC IDs, but to do so in a much shorter time period, allowing connections to be established in a fraction of the time they normally take. This method is useful in applications where nodes are added to an operating network, and the time required for establishing connections directly impacts productivity. For example, in robotic applications, the end-of-arm electronics are often changed out when a new item enters its workspace. These electronics need to be operational very quickly to avoid cycle time delays.

The QuickConnect process includes all the same steps as the typical startup process, but most of them are done in parallel rather than in sequence. As a result, the device self-check and Duplicate MAC ID Check processes begin immediately, and the node goes online almost simultaneously. A failure of the device self-test or a duplicate MAC ID indication causes the device to remove itself from the bus.

In order for applications to benefit fully from this method, QuickConnect must be implemented in both the master and the slave. This feature is selectable through an EDS entry, and by default, is disabled in nodes that support it.

3.1.17.8. Master/Slave Summary

Device manufacturers can easily support the predefined master/slave connection set by using simple BasicCAN controllers. Software screening of the CAN Identifier generally is not necessary, which enables the use of low-cost, 8-bit controllers. This may represent an advantage as far as the devices are concerned but entails some disadvantages for the system design.

Group 2 Only (i.e., UCMM incapable) devices permit only one explicit messaging connection between client (master) and server (slave), whereas UCMM capable devices can maintain explicit messaging connections with more than one client at the same time.

If a device wants to communicate with one of the allocated slaves that does not support UCMM, the master recognizes this situation and sets up a communication relationship with the requestor instead. Any communication between the requestor is then automatically routed via the master. This is called the Proxy function. Since this puts an additional burden on the master and on network bandwidth, it is recommended that slave devices support UCMM.

Although not explicitly defined in The DeviceNet Specification, DeviceNet masters can, under certain conditions, automatically configure their scan lists and/or the devices contained in their scan lists. This functionality simply makes use of the explicit messaging capabilities of masters and slaves that allows the master to read from a slave whatever information is required to start an I/O communication and to download any configurable parameters that have been communicated to the master via EDS. This functionality facilitates the replacement of even complex slave devices without the need for a tool, dramatically reducing system downtime.

3.1.18. Device Profiles

DeviceNet devices may utilize any of the device profiles described in the CIP Networks Library. As of the publication date of this book, no DeviceNet-specific profiles have been defined.

3.1.19. Configuration

DeviceNet devices typically come with Electronic Data Sheets (EDS) as described in chapter 2.7. EDS files for DeviceNet devices can make full use of all EDS features, but they do not necessarily contain all sections. Typical DeviceNet devices contain (apart from the mandatory sections) at least an IO_Info section.

This section specifies which types of master/slave connections are supported and which one(s) should be enabled as defaults. It also tells which I/O connections may be used simultaneously.

An EDS also may contain individual parameters and/or a configuration assembly with a complete description of all parameters within this assembly. A full description of what can be done in DeviceNet EDS files would go well beyond the scope of this book. For available materials on this topic that go into more detail see [41], [42].

3.1.20. Conformance Test

See chapter 6 of this publication for information on conformance testing.

3.1.21. Tools

Tools for DeviceNet networks can be divided into three groups:

- Physical layer tools are tools (hardware and/or software) that verify the integrity and conformance of the physical layer or monitor the quality of the data transmission;

- Configuration tools are software tools capable of communicating with individual devices for data monitoring and configuration purposes. They can range from very basic software operating on handheld devices to powerful PC-based software packages used to configure complete networks. Most configuration tools are EDS-based; however, more complex devices like I/O Scanners tend to have their own configuration applets that are only partially based on EDS files. Some of these tools support multiple access paths to the network, e.g., via Ethernet and suitable routing devices, and thus allow remote access. High-level tools also actively query the devices on the network to identify them and monitor their "health";

- Monitoring tools typically are PC-based software packages that can capture and display CAN frames on the network. A raw CAN frame display may be good enough for some experts, but using a tool that allows both raw CAN display and DeviceNet interpretation of the frames is recommended.

For a typical installation, a configuration tool is all that is needed. However, to ensure that the network is operating reliably, verification with a physical layer tool is highly recommended. Experience shows that the overwhelming majority of DeviceNet network problems are caused by inappropriate physical layer installation. Protocol monitoring tools are used primarily to investigate interoperability problems and to assist during the development process. Turn to the ODVA Marketplace on the ODVA website to access a list of vendors that provide tools for DeviceNet.

3.1.22. Advice for Developers

Before starting any DeviceNet product development, the following issues should be considered in detail:

- What functionality does the product require today and in future applications?
    - Slave functionality;
    - Group 2 Server vs. Group 2 Only Server.
    - Master functionality;
    - Combination of the above.
- What are the physical layer requirements? Is IP 65/67 required or is IP 20 good enough?
- What type of hardware should be chosen for this product?
- What kind of firmware should be used for this product? Will a commercially available communication stack be used?
- Will the development of hardware and/or software be done internally or will it be designed by an outside company?
- What are the configuration requirements?
- What design and verification tools should be used?
- What kind of configuration software should be used for this product? Will a commercially available software package be used, i.e., is an EDS adequate to describe the device or is custom software needed?
- When and where will the product be tested for conformance and interoperability?
- What is an absolute must before my products can be placed on the market (i.e., own the specification, have the company's own vendor ID, have the product conformance tested)?

A full discussion of these issues goes well beyond the scope of this publication, for more information see [43].

3.1.23. DeviceNet Summary

Since its introduction in 1994, DeviceNet has been used successfully in tens of millions of nodes in many different applications. It is a de facto standard in many countries, which is reflected in several national and international standards [25], [29], [30]. Due to its universal communication characteristics, it is one of the most versatile networks for low-end devices. While optimized for devices with small amounts of I/O, it can easily accommodate larger devices as well. Powerful EDS-based configuration tools allow easy commissioning and configuration of even complex devices without the need to consult manuals.

With the introduction of CIP Safety on DeviceNet, many machine-level applications that previously required a set of dedicated networks today can be accommodated on a single DeviceNet network.

Finally, as a member of the CIP family of networks, DeviceNet can be combined into an overall CIP Network structure that allows seamless communication among CIP Networks, as if they were only one network.

3.2. ControlNet

3.2.1. Introduction

Introduced in 1997, ControlNet is a deterministic digital communications network that provides high-speed transport of time-critical I/O and explicit messaging data – including upload/download of programming and configuration data and peer-to-peer messaging – on a single physical media link. Each device and/or controller is a node on the network.

ControlNet is a producer/consumer network that supports multiple communication hierarchies and message prioritization. ControlNet systems offer a single point of connection for configuration and control by supporting both implicit (I/O) and explicit messaging. ControlNet's time-based message scheduling mechanism provides network devices with deterministic and predictable access to the network while preventing network collisions. This scheduling mechanism allows time-critical data, which is required on a periodic, repeatable and predictable basis, to be produced on a predefined schedule without the loss of efficiency associated with continuously requesting, or "polling", for the required data.

3.2.2. Relationship to Standards

Like other CIP Networks, ControlNet follows the Open Systems Interconnection (OSI) model, an ISO standard for network communications that is hierarchical in nature. Networks that follow this model define all necessary functions, from physical implementation up to the protocol and methodology to communicate control and information data within and across networks.

Figure 26 shows the relationship between CIP and ControlNet.

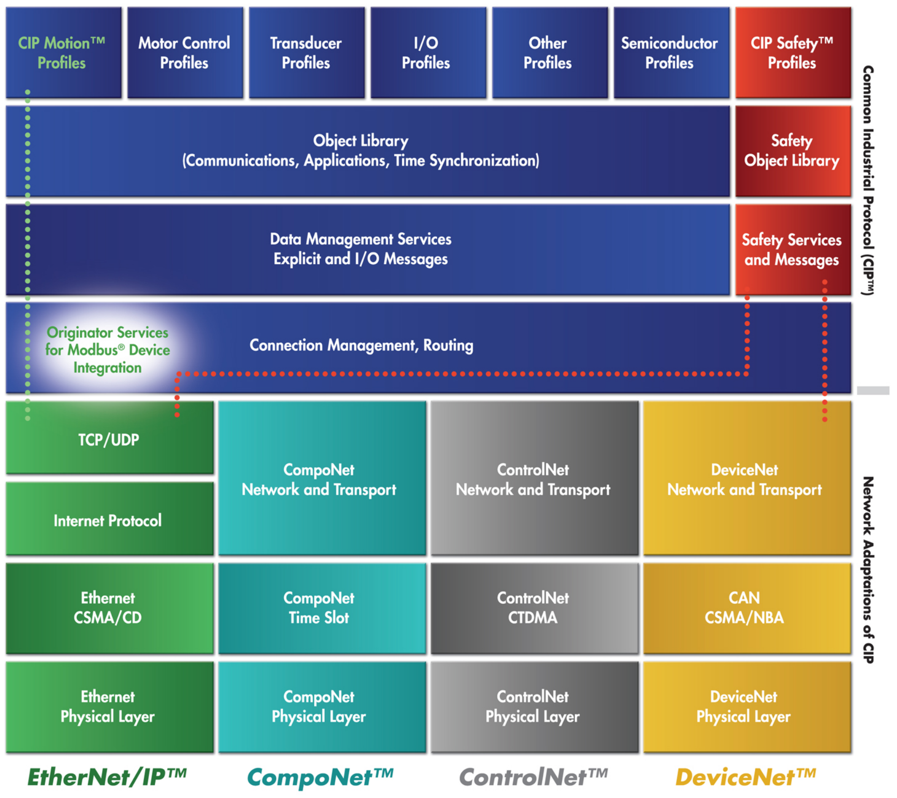

CIP Motion™ Profiles
Motor Control Profiles
Transducer Profiles
I/O Profiles
Other Profiles
Semiconductor Profiles
CIP Safety™ Profiles
Object Library (Communications, Applications, Time Synchronization)
Safety Object Library
Data Management Services Explicit and I/O Messages
Safety Services and Messages
Originator Services for Modbus® Device Integration
Connection Management, Routing
TCP/UDP
Internet Protocol
Ethernet CSMA/CD
Ethernet Physical Layer
CompoNet Network and Transport
CompoNet Time Slot
CompoNet Physical Layer
ControlNet Network and Transport
ControlNet CTDMA
ControlNet Physical Layer
DeviceNet Network and Transport
CAN CSMA/NBA
DeviceNet Physical Layer
EtherNet/IP™
CompoNet™
ControlNet™
DeviceNet™
Common Industrial Protocol (CIP™)
Network Adaptations of CIP

Figure 26 Relationship Between CIP and ControlNet

The ControlNet adaptation of CIP is described in Volume 4 of the CIP Networks Library [4]. All other features are based on CIP. ControlNet is also described in international standards, e.g., [31].

3.2.3. ControlNet Features

ControlNet is a high-speed deterministic industrial communication system with the following features:

- Trunkline/dropline configuration (copper media), star configuration (optical media);
- Support for media redundancy;
- Support for up to 99 nodes;
- Node insertion or removal while the network is up and running;
- Use of sealed or open-style connectors;
- Fixed baudrate (5 Mbaud).

3.2.4. ControlNet Physical Layer

The physical layer of ControlNet has been designed specifically for this network; it does not reuse any existing open technology. The basis of the physical layer is a 75 Ω coaxial trunkline (typically of RG-6 type cable) terminated at both ends with 75 Ω terminating resistors. To reduce impedance mismatch, all ControlNet devices are connected to the network through special taps that consist of a coupling network and a specific length of dropline (1 m). There is no minimum distance requirement between taps, but, since every tap introduces some signal attenuation, each tap reduces the maximum length of the trunkline by 16.3 m. This results in a full length trunkline of 1,000 m with only two taps at the ends while a fully populated physical network with 48 taps allows a trunkline length of 250 m (see Figure 27).

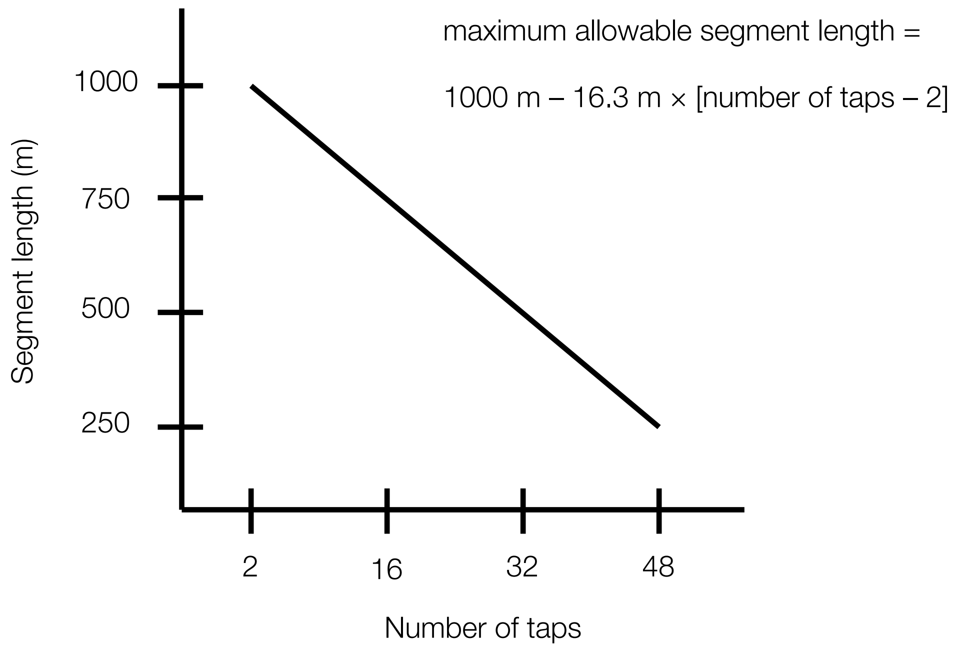


Figure 27 Coax Medium Topology Limits

This physical layer limitation is addressed by including repeaters that can increase the network size without lowering the speed. Therefore, if a network is to be built with a higher number of nodes (up to 99 nodes are possible) or with a topology that goes beyond the single trunkline limitations, repeaters can be used to extend the bus. It's possible to create any type of topology: tree, star or linear bus. Even a ring topology is possible using a special type of repeater. Repeaters for fiber optic media can be used either to further increase the system size or to allow isolation of network segments in harsh EMC environments or for high voltage applications.

The number of repeaters between any two nodes was initially limited to five, but further technology developments now allow up to 20 repeaters in series. However, regardless of the media technology used, the overall length of a ControlNet system (the distance between any two nodes on the network) is still limited by propagation delay. With currently available media, this translates into approximately 20 km.

To better accommodate industry requirements, ControlNet supports redundant media, allowing bumpless transfer from primary to secondary media or vice versa if one of them should fail or deteriorate. Developers are encouraged to support this redundant media feature in their designs. For cost-sensitive applications, less expensive device variants may then be created by populating one channel only.

Another feature often used in the process industry is the capability of running ControlNet systems into areas with an explosion hazard. ControlNet is fully approved to meet worldwide standards for intrinsic safety (explosion protection).

Copper media uses BNC type connectors for IP 20 type applications and TNC type connectors for IP 67 protection. Devices also may implement a network access port (NAP). This feature takes advantage of the repeater function of the ControlNet ASICs. It uses an additional connector (RJ-45) with RS 422-based signals that provides easy access to any node on the network for configuration devices.

The signal transmitted on the copper media is a 5-Mbit/s Manchester-encoded signal with an amplitude of up to 9.5 V (pk-pk) at the transmitter that can be attenuated down to 510 mV (pk-pk) at the receiving end. The specification provides reference transmitting and receiving circuits.

3.2.5. Frame Structure

Every frame transmitted on ControlNet has the format of the MAC frame shown in Figure 28.

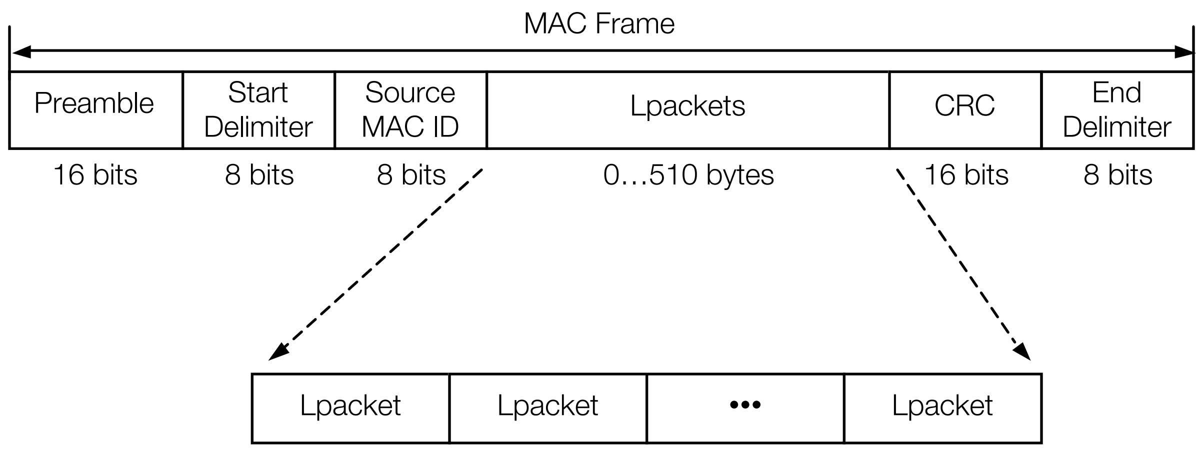


Figure 28 MAC Frame Format

Within every MAC frame, a field of up to 510 bytes is available for transmitting data or messages. This field may be populated with one or several Lpackets (link packets). These Lpackets carry the individual CIP messages (I/O or Explicit). Specialized Lpackets are used for network management. Since all nodes always listen to all MAC frames, they have no problem consuming any of the Lpackets in a frame that is unicast, multicast or broadcast in nature. This feature allows fine-tuned multicasting of small amounts of data to different sets of consumers without much overhead.

There are two types of Lpacket formats: fixed tag and generic tag. Fixed tag Lpackets are used for unconnected messaging and network administration packets, while the generic tag Lpackets are used for all connected messaging (I/O and Explicit).

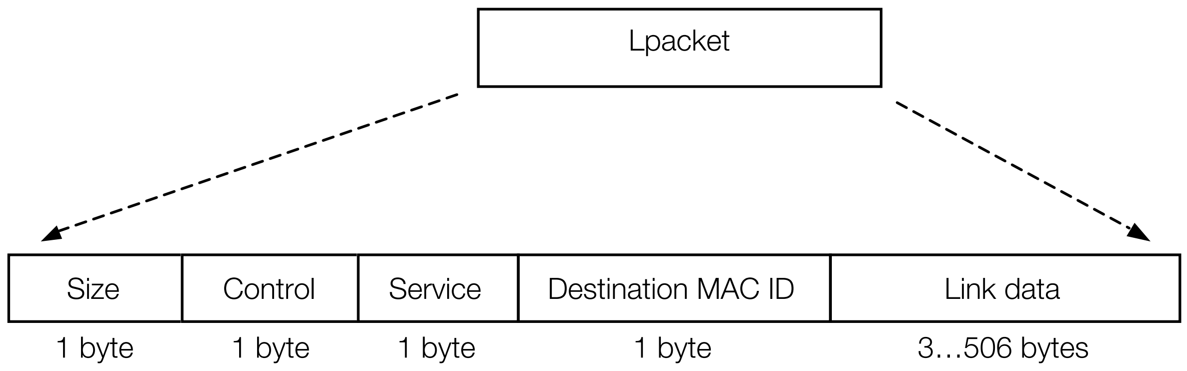


Figure 29 Fixed Tag Lpacket Format

Figure 29 shows the format of a fixed tag Lpacket. By including the destination MAC ID, this format reflects the fact that these Lpackets are always directed from the requesting device (sending the MAC frame) to the target device (the destination MAC ID). The service byte within a fixed tag Lpacket does not represent the service of an explicit message, but a service type on a different level, since the fixed tag Lpacket format can be used for a variety of actions, such as network administration.

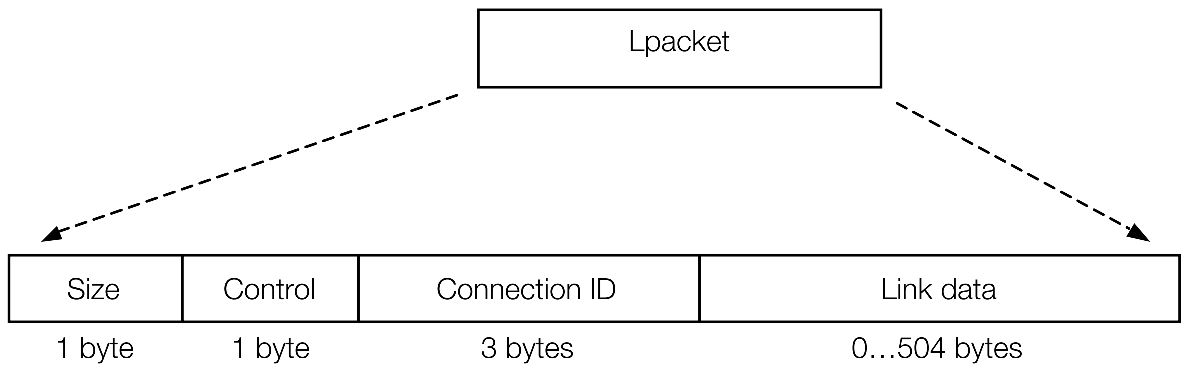


Figure 30 Generic Tag Lpacket Format

Figure 30 shows the format of a generic tag Lpacket. The size byte specifies the number of words within the Lpacket, while the control byte gives information on what type of Lpacket this is. The 3-byte Connection Identifier specifies which connection this Lpacket belongs to. These three bytes are the three lower bytes of the 4-byte connection ID specified in the Forward_Open message; the uppermost byte is always zero. For a device that receives the MAC frame, the connection ID indicates whether to ignore the Lpacket (the device is not part of the connection), to consume the data and forward it to the application (the device is an end point of this connection) or to forward the data to another network (the device acts as a router in a routed connection).

3.2.6. Protocol Adaptation

ControlNet can use all features of CIP. The ControlNet frame is big enough that fragmentation is rarely required and thus is only provided by application-specific services that might require it. Since ControlNet is not used in very simple devices, no scaling is required.

3.2.7. Indicators and Switches

ControlNet devices must be built with Device Status and Network Status indicators as described in the specification. Devices may have additional indicators which must not carry any of the names of those described in the specification.

Devices may be built with or without switches or other directly accessible means for configuration. If switches for the MAC ID exist, then certain rules apply regarding how these values must be used at power up and during the operation of the device.

3.2.8. Additional Objects

Volume 4 defines three additional objects: the ControlNet object, the keeper object and the scheduling object.

3.2.8.1. ControlNet Object (Class ID: 0xF0)

The ControlNet object contains a host of information about the state of the device's ControlNet interface, among them diagnostic counters, data link and timing parameters and the MAC ID. A ControlNet object is required for every physical layer attachment of the device. A redundant channel pair counts as one attachment.

3.2.8.2. Keeper Object (Class ID: 0xF1)

The keeper object (not required for every device) holds (for the network scheduling software) a copy of the Connection Originator schedule data for all Connection Originator devices using the network. Every ControlNet Network with scheduled I/O traffic must have at least one device with a keeper object (typically, a PLC or another Connection Originator). If there are multiple keeper objects on a network, they perform negotiations to determine which Keeper is the Master Keeper and which Keeper(s) perform Backup Keeper responsibilities. The Master Keeper is the Keeper actively distributing attributes to the nodes on the network. A Backup Keeper is one that monitors Keeper-related network activity and can transition into the role of Master Keeper should the original Master Keeper become inoperable.

3.2.8.3. Scheduling Object (Class ID: 0xF2)

The scheduling object is required in every device that can originate an I/O messaging connection. Whenever a network scheduling tool accesses a Connection Originator on a ControlNet Network, an instance of the scheduling object is created and a set of object-specific services is used to interface with this object. Once the instance is created, the network scheduling tool can then read and write connection data for all connections that originate from this device. After having read the connection data from all Connection Originators, the network scheduling tool can calculate an overall schedule for the ControlNet Network and write this data back to all Connection Originators. The scheduling session is ended by deleting the instance of the scheduling object.

3.2.9. Network Access

ControlNet's bus access mechanism allows full determinism and repeatability while still maintaining sufficient flexibility for various I/O message triggers and explicit messaging. This bus access mechanism is called Concurrent Time Domain Multiple Access (CTDMA); it is illustrated in Figure 31.

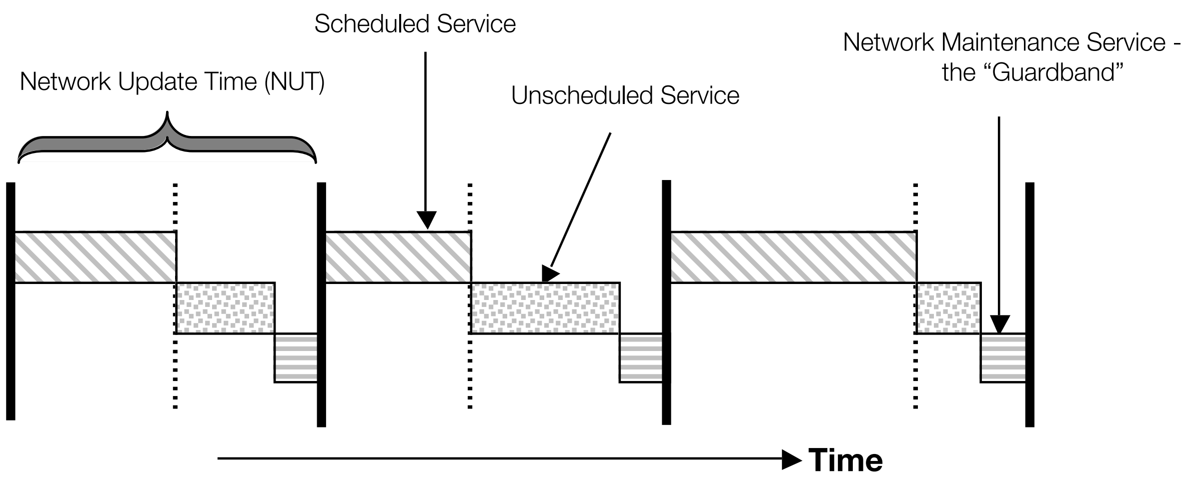

Scheduled Service
Network Update Time (NUT)
Unscheduled Service
Network Maintenance Service - the "Guardband"
Time

Figure 31 Media Access through CTDMA (Concurrent Time Domain Multiple Access)

The time axis is divided into equal intervals called Network Update Time (NUT). Each NUT is subdivided into the Scheduled Service Time, the Unscheduled Service Time and the Guardband Time.

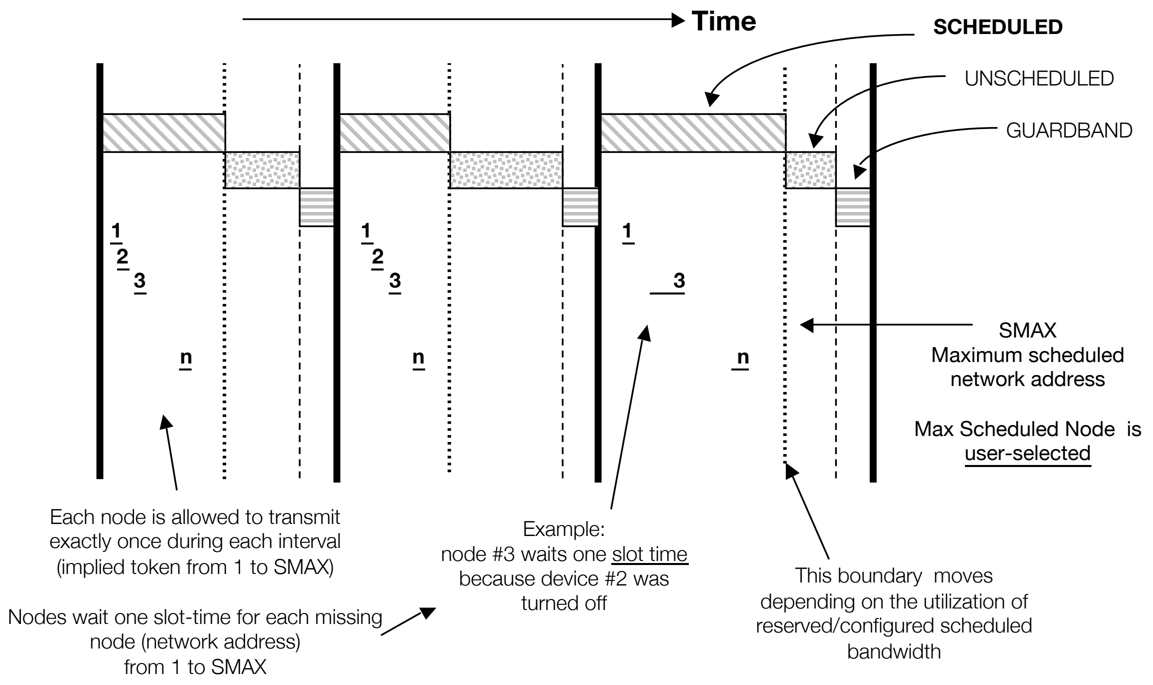

Time
SCHEDULED
UNSCHEDULED
GUARDBAND
1
2
3
n
SMAX
Maximum scheduled network address
Max Scheduled Node is user-selected
Each node is allowed to transmit exactly once during each interval (implied token from 1 to SMAX)
Nodes wait one slot-time for each missing node (network address) from 1 to SMAX
Example: node #3 waits one slot time because device #2 was turned off
This boundary moves depending on the utilization of reserved/configured scheduled bandwidth

Figure 32 Scheduled Service

Figure 32 shows the function of the Scheduled Service. Every node up to, and including, the SMAX node (maximum node number participating in the Scheduled Service) has a chance to send a message within the Scheduled Service. If a particular node has no data to send, it will send a short frame to indicate that it is still alive. If a node fails to send its frame, the next-higher node number will step in after a very short, predetermined waiting time. This process ensures that a node failure will not lead to an interruption of the NUT cycle.

Figure 33 shows the function of the Unscheduled Service. Since this service is designed for non-time-critical messages, only one node is guaranteed access to the bus during the Unscheduled Service Time. If there is time left, the other nodes (with higher node numbers) will also get a chance to send. As with the Scheduled Service Time, if a node fails to send during its turn, the next node will step in. The node number that is allowed to send first within the Unscheduled Service Time is increased by one in each NUT. This guarantees an equal chance to all nodes. When the node sequencing within a NUT reaches the maximum value, known as UMAX, it wraps around to node 1, and the sequence resumes.

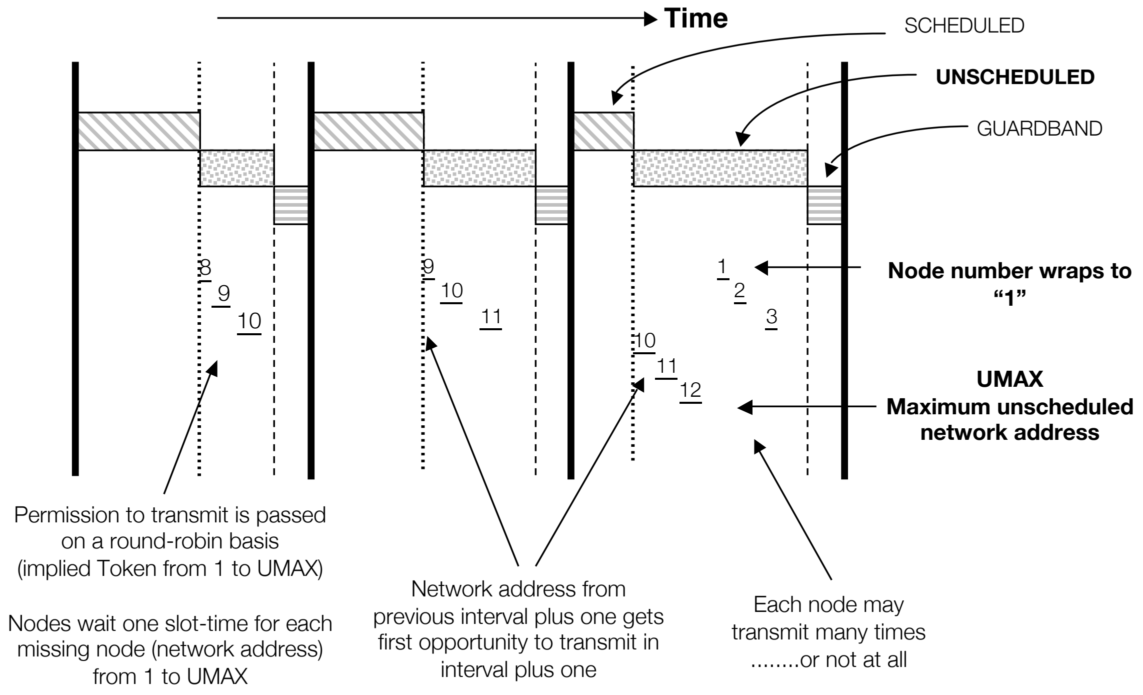

Time
SCHEDULED
UNSCHEDULED
GUARDBAND
8
9
10
9
10
11
10
11
12
1
2
3
Node number wraps to "1"
UMAX
Maximum unscheduled network address
Permission to transmit is passed on a round-robin basis (implied Token from 1 to UMAX)
Nodes wait one slot-time for each missing node (network address) from 1 to UMAX
Network address from previous interval plus one gets first opportunity to transmit in interval plus one
Each node may transmit many times ....or not at all

Figure 33 Unscheduled Service

These two service intervals, combined with the Guardband, guarantee determinism and repeatability while still maintaining sufficient freedom to allow for unscheduled message transmissions, e.g., for parameterization.

3.2.10. Network Startup

After power-on, every ControlNet device goes through a process of accessing the ControlNet communication network and learning the current NUT and other timing requirements. This is a fairly complex process typically handled by commercially available ControlNet ASICs. It is beyond the scope of this book to describe the details here.

3.2.11. Explicit Messaging

Explicit messages on ControlNet, unlike those on DeviceNet, can be sent either connected or unconnected; both are transmitted within the unscheduled part of the NUT. Connected explicit messaging requires setting up a connection (see chapter 3.2.13) before messages are exchanged. This means that all resources required for managing the connection are reserved for this purpose as long as the connection exists, which allows more timely responses to message requests. This is very useful when the application requires periodic explicit requests. Most explicit messages also can be sent unconnected, but this mechanism makes use of generally limited resources in nodes that sometimes can be highly utilized. For this reason, unconnected explicit messaging should be used only when the application requires very irregular and infrequent request intervals. Every part of an explicit message (request, response, acknowledgements) is wrapped into an Lpacket using the fixed tag Lpacket format for unconnected explicit messaging (see Figure 29) and the generic tag Lpacket format for connected explicit messaging (see Figure 30). The service/class/instance/attribute fields (see chapter 2.3) of the explicit message are contained in the link data field.

3.2.12. I/O Messaging

ControlNet I/O messaging is accomplished using connections, and always takes place in the scheduled part of the NUT. Only one MAC frame may be transmitted by any device within its time slot, but this MAC frame may contain multiple Lpackets so that data can be sent to multiple nodes in one NUT. The individual Lpackets may be consumed by one node only or by multiple nodes if they are set up to consume the same data.

I/O messages use the generic tag Lpacket format (see Figure 30). The link data field contains the I/O data prefixed with a 16-bit sequence count value for the packet. Run/Idle can be indicated within a prefixed Real-Time Header or by sending the data packet (Run) or no data packet (Idle). The method used is indicated in the connection parameters of the Connection Manager section of the EDS. However, only the Real-Time Header method has been used for ControlNet up to now.

3.2.13. Connection Establishment

All connections on ControlNet are established using a UCMM Forward_Open message (see Section 2.3).

3.2.14. Device Classes

Four classes of device functionality are built with CIP. While they are not explicitly defined in the specification, they are useful for distinguishing among several classes of devices. The four classes are described here:

- The minimal device function is that of an explicit message Server, which is used for explicit messaging applications only and acts as a target for Unconnected and (optionally) connected explicit messages, e.g., for program upload/download, data collection, status monitoring, etc.;

- The next device class is an I/O Server, which adds I/O messaging support to an explicit message Server device and acts as a target for both Explicit and I/O messages, e.g., simple I/O Devices, Pneumatic Valves, AC Drives. These devices are also called I/O Adapters;

- Another device class is an explicit message client, which adds client support to explicit message Server applications and acts as a target and as an originator for explicit messaging applications, e.g., computer interface cards, HMI devices;

- The most powerful type of device is an I/O Scanner, which adds I/O message origination support to the functionality of all the other device classes, and which acts as a target and as an originator for Explicit and I/O messages, e.g., PLCs, I/O Scanners.

3.2.15. Device Profiles

ControlNet devices may utilize any device profiles described in the CIP Networks Library. As of the publication date of this book, no ControlNet-specific profiles have been defined.

3.2.16. Configuration

ControlNet devices typically come with Electronic Data Sheets (EDS) as described in Section 2.6. For EDS-based configuration tools, the EDS should contain a Connection Manager section to describe the details of the connections that can be made into the device. This section basically mirrors the contents of the Forward_Open message that a Connection Originator would send to the device. Multiple connections can be specified within an EDS, then one or more can be chosen by the configuration tool.

An EDS may also contain individual parameters and/or a configuration assembly with a complete description of all parameters within this assembly. In many applications, the configuration assembly is transmitted as an attachment to the Forward_Open message.

3.2.17. Conformance Test

See chapter 6 of this publication for information on conformance testing.

3.2.18. Tools

Tools for ControlNet Networks can be divided into three groups:

- Physical layer tools are tools (hardware and/or software) that verify the integrity and conformance of the physical layer or monitor the quality of the data transmission;

- Configuration tools are software tools capable of communicating with individual devices for data monitoring and configuration purposes. Most configuration tools are EDS-based; however, more complex devices like I/O Scanners tend to have their own configuration applets that are only partially based on EDS files. Some of these tools support multiple access paths to the network, e.g., via Ethernet and suitable routing devices, and thus allow remote access. High-level tools also actively query the devices on the network to identify them and monitor their health. Configuration tools also may be integrated into other packages like PLC programming software;

- Monitoring tools typically are PC-based software packages that can capture and display the ControlNet frames on the network. A raw ControlNet frame display may be good enough in some instances, but using a tool that can display both raw ControlNet frames and interpreted frames is recommended.

For a typical installation, a configuration tool is all that is needed. However, to ensure the network is operating reliably, testing with a physical layer tool is highly recommended. Experience shows that the overwhelming majority of ControlNet network problems are caused by inappropriate physical layer installation. Protocol monitoring tools are mainly used to investigate interoperability problems and to assist during the development process.

Turn to the ODVA Marketplace on the ODVA website to access a list of vendors that provide tools for ControlNet.

3.2.19. Advice for Developers

Before any development of a ControlNet product is started, the following issues should be considered in detail:

- What functionality (Device Classes, see chapter 3.2.14) does the product require today and in future applications?
    - Explicit messaging server only;
    - I/O Adapter functionality;
    - Explicit messaging client;
    - I/O scanner functionality.
- What are the physical layer requirements? Is IP 65/67 required or is IP 20 good enough?
- Will the development be based on commercially available hardware components and software packages (recommended) or designed from scratch (possible but costly)?
- What are the configuration requirements?
- What design and verification tools should be used?
- When and where will the product be tested for conformance and interoperability?
- What is an absolute must before products can be placed on the market (own the specification, have the company's own vendor ID, have the product conformance tested)?

Rockwell Automation has published a comprehensive developer's handbook that assists vendors in developing products, see [44]. ControlNet chipsets and associated software packages are available from Rockwell Automation. Turn to the ODVA website for a list of companies that can support ControlNet developments.

3.2.20. ControlNet Summary

Since its introduction in 1997, ControlNet has been used successfully in millions of nodes in many different applications. It is the network of choice for many high speed I/O and PLC interlocking applications. Like DeviceNet, ControlNet has become an international standard [31]. Due to its universal communication characteristics, it is one of the most powerful controller-level networks available.

ControlNet's greatest strengths are its media redundancy and its full determinism and repeatability. These strengths make it ideally suited for many applications that require media redundancy and also for many high-speed applications, in which ControlNet maintains full explicit messaging capabilities without compromising its real-time behavior.

Finally, as a member of the CIP family of networks, ControlNet can be combined into an overall CIP Network structure that allows seamless communication among CIP Networks, just as if they were only one network.

3.3. EtherNet/IP

3.3.1. Introduction

Introduced in 2000, EtherNet/IP is another member of the CIP Family. Using CIP as its upper-layer protocol, EtherNet/IP extends the application of Ethernet TCP/IP to the plant floor. EtherNet/IP can coexist with any other protocol running on top of the standard TCP/UDP Transport Layer, and with other CIP Networks. EtherNet/IP – CIP plus Internet and Ethernet standards – provides a pure, unmodified, standards-based Ethernet solution for interoperability among manufacturing enterprise networks, and it enables Internet and enterprise connectivity anywhere, anytime utilizing commonly available switches. The “IP” in EtherNet/IP stands for the “Industrial Protocol” in CIP; this is not to be confused with “IP” in TCP/IP which stands for “Internet Protocol”.

Due to the length of Ethernet frames and the typical multi-master structure of Ethernet networks, there are no particular limitations in the EtherNet/IP implementation of CIP. Basically, all that is required is a mechanism to encode CIP messages into Ethernet frames.

3.3.2. Relationship to standards

Like other CIP Networks, EtherNet/IP follows the Open Systems Interconnection (OSI) model, an ISO standard for network communications that is hierarchical in nature. Networks that follow this model define all necessary functions, from physical implementation up to the protocol and methodology to communicate control and information data within and across networks.

Figure 34 shows the relationship between CIP and EtherNet/IP.

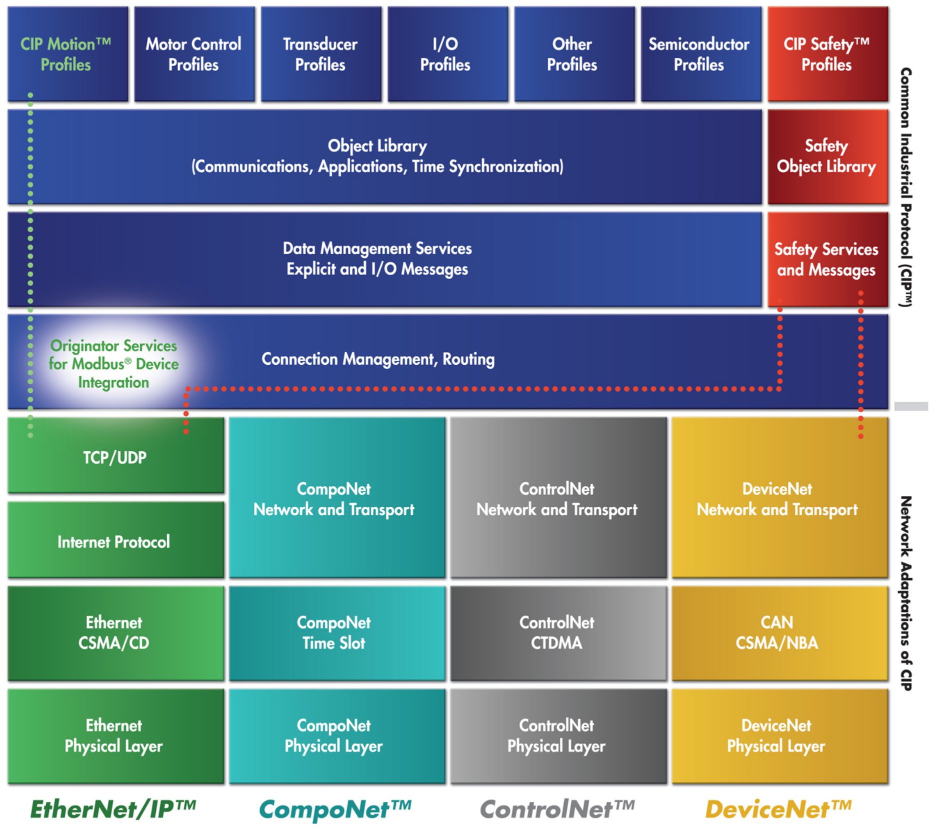

CIP Motion™ Profiles
Motor Control Profiles
Transducer Profiles
I/O Profiles
Other Profiles
Semiconductor Profiles
CIP Safety™ Profiles
Object Library (Communications, Applications, Time Synchronization)
Safety Object Library
Data Management Services Explicit and I/O Messages
Safety Services and Messages
Originator Services for Modbus® Device Integration
Connection Management, Routing
Common Industrial Protocol (CIP™)
TCP/UDP
Internet Protocol
CompoNet Network and Transport
ControlNet Network and Transport
DeviceNet Network and Transport
Ethernet CSMA/CD
CompoNet Time Slot
ControlNet CTDMA
CAN CSMA/NBA
Ethernet Physical Layer
CompoNet Physical Layer
ControlNet Physical Layer
DeviceNet Physical Layer
Network Adaptations of CIP
EtherNet/IP™
CompoNet™
ControlNet™
DeviceNet™

Figure 34 Relationship Between CIP and EtherNet/IP

The EtherNet/IP adaptation of CIP is described in Volume 2 of the CIP Networks Library. All other features are based on CIP. This volume defines how CIP is adapted for use on Ethernet. An encapsulation mechanism (see chapter 3.3.11) is defined for EtherNet/IP specifying how I/O and explicit messages are carried in Ethernet frames. The well-known TCP/IP protocol is used for encapsulating explicit messages, while UDP/IP is used for encapsulating I/O messages. Since the commonly implemented TCP/IP and UDP/IP protocol stacks are used for encapsulation, many applications will not require extra middleware for this purpose.

Ethernet has its roots in the office computing environment, which is not traditionally concerned with determinism like industrial applications are. However, with the proper selection and configuration of infrastructure devices (see chapter 3.3.21) using fast data rates with full duplex communications there will be no collisions or lost packets, giving Ethernet a level of determinism that is more than adequate for use in industrial control applications. Additionally, extensions to CIP like CIP Sync and CIP Motion (see Section 5.1) allow EtherNet/IP to be used in highly synchronous and deterministic applications like coordinated drives and motion control.

EtherNet/IP is also described in international standards, i.e., the IEC fieldbus standards, see [27], [31].

3.3.3. EtherNet/IP Features

EtherNet/IP is a communication system built on standard, unmodified Ethernet with the following features:

- Built on and compliant with the relevant Ethernet standards, not just compatible with them;
- Fully independent of data rate: 10, 100, 1000 Mbit/s;
- Systems can be built with standard infrastructure;
- Virtually unlimited number of nodes in a network;
- Networks can be structured into subnets with IP routers;
- Full support of communication across subnets since EtherNet/IP uses IP addressing for all communication;
- Non-realtime communication and realtime communication can coexist in the same subnet;
- Support for coordinated drives and motion control;
- Support for Device Level Ring (DLR) which provides single fault tolerance through media redundancy;
- QuickConnect for devices that are frequently removed from and added to the network, e.g., robot tools;
- Coexistence with other upper-layer protocols, such as HTTP, FTP, VOIP etc.

3.3.4. EtherNet/IP Physical Layer

Since EtherNet/IP takes the Ethernet protocol to the factory floor, recommendations are made in Volume 2 [2] regarding grounding, isolation and cable and connector construction that are designed to make EtherNet/IP successful in a typical factory automation environment. These changes do not affect the actual signaling or interoperability with standard Ethernet products, but simply make devices more suitable for harsher industrial environments. As a result, two levels of performance criteria are defined:

- The COTS (Commercial Off The Shelf) EtherNet/IP Level provides basic Ethernet connectivity. This level includes the well-known RJ-45 type Ethernet connector but specifies topology constraints (e.g., up to 100 m) and cabling requirements through references to specific IEEE, ANSI/TIA/EIA standards. Such devices are typically suited for IP 20 applications;

- The Industrial EtherNet/IP Level goes beyond the COTS Level by specifying minimum environmental, cabling and connector requirements that include IEC, ANSI/TIA/EIA standards. Connectors required for the Industrial EtherNet/IP Level include an enhanced performance RJ-45 connector, a sealed RJ-45 connector as well as a more compact, D-coded M 12-4 connector. The sealed RJ-45 and M 12 connectors can achieve an IP 67 rating.

Cat 5E or Cat 6 shielded or unshielded cables are recommended for EtherNet/IP. The use of shielded cables is specifically recommended in applications where adjacent material, such as metal cable ducts, may have substantial influence on the characteristics of the cable. In accordance with IEEE 802.3, copper media may be used only for distances up to 100 m. Fiber-optic media is recommended for longer distances. Fiber-optic media may also be advisable for applications with very high electromagnetic disturbances or high-voltage potential differences between devices.

ODVA has published a guideline for the installation of Ethernet media, see [15]. This topic is also covered by the international standard IEC 61784-5-2 [27].

3.3.5. Frame Structure

EtherNet/IP uses standard Ethernet TCP/IP and UDP/IP frames as defined by international standards [22], [47], [48], [50]. Therefore, no further frame details are described here.

3.3.6. Protocol Adaptation

EtherNet/IP can use all features of CIP. The Ethernet frame is big enough that fragmentation is rarely required. If it is required, fragmentation is automatically handled by IP fragmentation provided by TCP/IP and UDP/IP. Since EtherNet/IP is not expected to be used in very simple devices, no further scaling is required.

3.3.7. Indicators and Switches

EtherNet/IP devices that need to conform to the Industrial EtherNet/IP Level must have the two indicators set forth in the specification: Module Status and Network Status. Devices may have additional indicators which must not carry any of the names of those described in the specification.

Devices may be built with or without switches or other manual means for configuration.

3.3.8. Additional Objects

Volume 2 defines the following 11 additional objects that are found only on EtherNet/IP devices. Most of these objects are only required when the feature they pertain to are implemented. Exceptions to this are noted where appropriate.

3.3.8.1. TCP/IP Interface Object (Class ID: 0xF5)

The TCP/IP interface object provides a mechanism for configuring a device's TCP/IP network interface. Examples of configurable items include the device's IP address, network mask and gateway address. Every EtherNet/IP must have at least one instance of this class.

3.3.8.2. Ethernet Link Object (Class ID: 0xF6)

The Ethernet Link object maintains configuration parameters, various error counters and status information for the Ethernet IEEE 802.3 communications interface. Each device has exactly one instance of the Ethernet Link object for each Ethernet IEEE 802.3 communications interface.

3.3.8.3. Device Level Ring (DLR) Object (Class ID: 0x47)

The DLR object manages all data and behavior associated with the DLR functionality of a device. For further details on DLR see chapter 3.3.23.2 of this publication.

3.3.8.4. QoS Object (Class ID: 0x48)

The QoS object manages all data and behavior associated with the QoS (Quality of service) functionality of a device. It includes the settings for DSCP in the IP header and the Frame Prioritization settings for the Ethernet header. If the device supports DLR, then this class must be implemented, too.

3.3.8.5. Base Switch Object (Class ID: 0x51)

The base switch object provides the CIP application-level interface to basic status information for a Managed Ethernet switch device. Devices shall implement no more than one instance of the base switch object.

3.3.8.6. Simple Network Management (SNMP) Object (Class ID: 0x52)

The SNMP object provides parameters used to configure aspects of the SNMP Agent in the device.

3.3.8.7. Power Management Object (Class ID: 0x53)

The power management object defines a Sleeping state and a Paused state. The method to trigger the transition from the Sleeping stat to the Paused state is network adaptation-specific.

3.3.8.8. RSTP Bridge Object (Class ID: 0x54)

The RSTP bridge object provides the configuration and diagnostic interface for the RSTP protocol at the bridge level. For further details on the use of RSTP, see Section 3.3.24 of this publication

3.3.8.9. RSTP Port Object (Class ID: 0x55)

The RSTP port object provides a configuration and diagnostic interface for the RSTP protocol at the port level. For further details on the use of RSTP, see Section 3.3.24 of this publication.

3.3.8.10. Parallel Redundancy Protocol (PRP) Object (Class ID: 0x56)

The Parallel Redundancy Protocol (PRP) object provides a configuration and diagnostic interface for PRP parameters, if implemented in the product.

3.3.8.11. PRP Nodes Table Object (Class ID: 0x57)

The PRP node table object keeps the record of all PRP capable nodes that have been detected on the network.

3.3.9. IP Address Assignment

Since the initial development of TCP/IP, numerous methods for configuring a device's IP address have evolved. Not all of these methods are suitable for industrial control devices. In the office environment, for example, it is common for a PC to obtain its IP address via DHCP (Dynamic Host Configuration Protocol), meaning that it can potentially acquire a different address each time the PC reboots. This is acceptable because the PC is typically a client device that only makes requests, so there is no impact if its IP address changes.

However, for an industrial control device that is a target of communication requests, the IP address cannot change at each power up. A PLC, for example, must be at the same address each time it powers up.

In addition, the only interface common to all EtherNet/IP devices is an Ethernet communications port. Some devices may also have a serial port, a user interface display, hardware switches or other interfaces, but these are not universally shared across all devices. Since Ethernet is the common interface, the initial IP address should at least be configurable over Ethernet.

The EtherNet/IP Specification, via the TCP/IP interface object, defines a number of ways to configure a device's IP address. A device may obtain its IP address via BOOTP (Bootstrap Protocol), via DHCP or via an explicit Set_Attribute_Single or Set_Attributes_All service. None of these methods is mandated however. As a result, vendors could choose different methods for configuring IP addresses.

From the user's perspective, it is desirable for vendors to support some common mechanism(s) for IP address configuration. The current ODVA recommendations on this subject can be downloaded from the ODVA website [9].

3.3.10. Address Conflict Detection (ACD)

Since IP addresses are often assigned by human interaction or as a default private address by the device manufacturer (e.g., 192.168.1.1), it is not uncommon to find multiple devices on the same network with the same IP address. This situation is undesirable; therefore, duplicate IP address detection and the subsequent address conflict resolution has been defined for EtherNet/IP. The ACD mechanism deployed in EtherNet/IP conforms to the IETF RFC 5227 [62]. Support for ACD is optional; however, any EtherNet/IP device that supports it, must follow the method described in Volume 2, Appendix F of The EtherNet/IP Specification.

3.3.11. EtherNet/IP Encapsulation

EtherNet/IP is based entirely on existing TCP/IP and UPD/IP technologies and uses them without any modification. TCP/IP is mainly used for the transmission of explicit messages while UDP/IP is used mainly for I/O messaging.

The encapsulation protocol defines two reserved TCP/UDP port numbers. All EtherNet/IP devices accept at least 2 TCP connections on TCP port number 0xAF12. This port is used for all TCP-based explicit messaging, either connected or unconnected. It is also used for the encapsulation protocol commands that are employed when setting up communications between nodes. Some encapsulation commands may also be sent to port 0xAF12 via UDP datagrams.

Port 0x08AE is used by any devices that support EtherNet/IP's I/O messaging over UDP. These messages can be sent either unicast or multicast by taking advantage of the multicast capabilities of IP. Multicast data flow makes more efficient use of the available bandwidth and provides for better data consistency across the system. Being connectionless, UDP is well suited to this purpose as connection management is handled by CIP.

3.3.11.1. General Use of the Ethernet frame

Since EtherNet/IP is completely based on Ethernet with TCP/IP and UDP/IP, all CIP-related messages sent on an EtherNet/IP network are Ethernet frames with an IP header (see Figure 35).

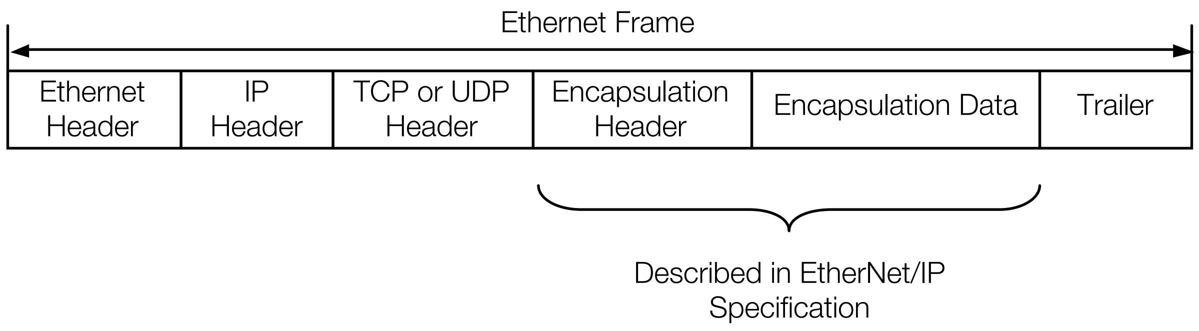


Figure 35 Relationship Between CIP and Ethernet Frames

The Ethernet header, the IP header and the TCP or UDP headers are described through international standards (see chapter 3.3.5); therefore, details of these headers are mentioned only in The EtherNet/IP Specification when it is necessary to understand how they are used to carry CIP.

The encapsulation header contains a command which determines the meaning of the encapsulation data. Many commands specify the use of the so-called Common Packet Format. I/O messages sent in UDP frames do not use the encapsulation header, but they still follow the Common Packet Format.

3.3.11.2. Encapsulation Header and Encapsulation Commands

The overall encapsulation packet has the structure described in Figure 36.

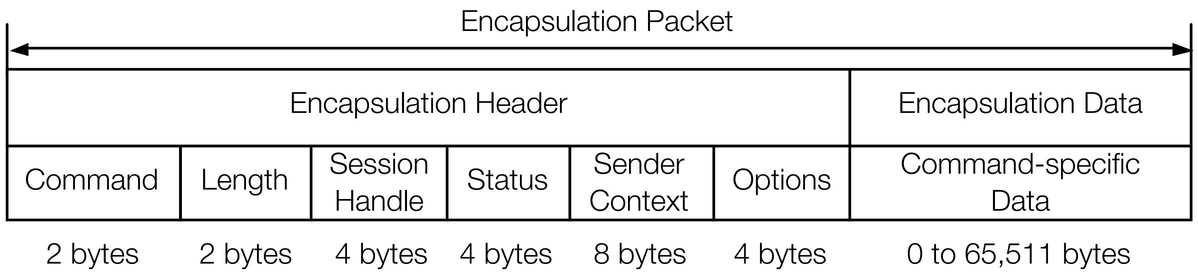


Figure 36 Structure of the Encapsulation Packet

While the description of some of the encapsulation header details would go beyond the scope of this book, the command field requires more attention here. However, only those commands that are needed to understand the EtherNet/IP protocol are described, and their description only lists the main features.

3.3.11.2.1. ListIdentity Command

The ListIdentity command typically is sent as a broadcast UDP message to tell all EtherNet/IP devices to return a data set with identity information. This command is used by software tools to browse a network.

3.3.11.2.2. RegisterSession/UnRegisterSession Commands

These two commands are used to open and close an Encapsulation Session between two devices. Once a Session is established, it is used to exchange more messages. Only one Session may exist between two devices.

The device receiving the RegisterSession request creates a Session Handle that it returns in the RegisterSession reply. This value is used to identify messages sent between the two devices that use this Session.

3.3.11.2.3. SendRRData/SendUnitData Commands

The SendRRData Command is used for unconnected explicit messaging, and the SendUnitData Command is used for connected explicit messaging. The device transmitting the SendRRData request creates a Sender Context value that is returned with the reply. The SendUnitData does not use the Sender Context field.

3.3.11.2.4. Common Packet Format

The Common Packet Format is a construct that provides a way to structure the Encapsulation Data field for those Encapsulation commands that specify Encapsulation data. The Common Packet Format defines Items that represent different types of information to be exchanged between devices. If the command definition requires it, the Common Packet Format allows packing of multiple Items into one encapsulation frame, as shown in Figure 37.

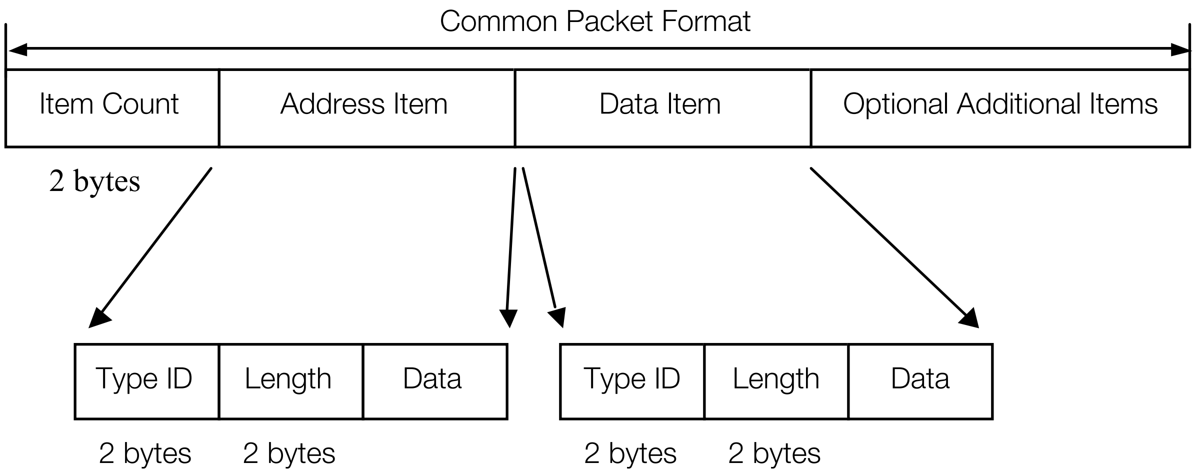


Figure 37 Example of the Common Packet Format

3.3.12. Use of the Encapsulation Data

3.3.12.1. Explicit Messaging

Explicit messages on EtherNet/IP can be sent either connected or unconnected. Connected explicit messaging requires setting up a connection (see chapter 3.3.13) before messages are exchanged. This means that all resources required for managing the connection are reserved for this purpose as long as the connection exists, which allows for more timely responses to message requests. This is very useful in applications that require periodic explicit requests. Explicit messages also can be sent unconnected, but this mechanism makes use of generally limited resources in nodes that sometimes can be highly utilized. For this reason, unconnected explicit messaging should be used only when the application requires very irregular and infrequent request intervals. Explicit messages on EtherNet/IP are sent with a TCP/IP header and use the SendRRData Encapsulation Command (unconnected) or the SendUnitData Encapsulation Command (connected). As an example, the full encapsulation of a UCMM request is shown in Figure 38.

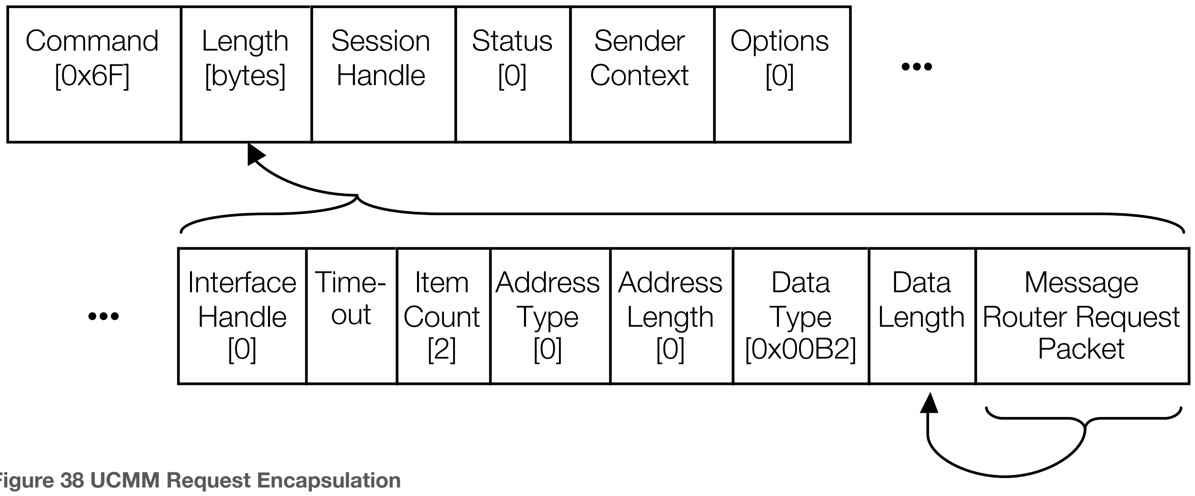


Figure 38 UCMM Request Encapsulation

The Message Router Request Packet noted in the figure contains the CIP message request or response. This part of the packet follows the general format of explicit messages – the Message Router Request/Response Format – defined in Volume 1, Chapter 2 of the CIP Networks Library.

3.3.12.2. I/O Messaging

I/O messages on EtherNet/IP are sent with a UDP/IP header. No encapsulation header is required, but the message still follows the Common Packet Format. See Figure 39 for an example.

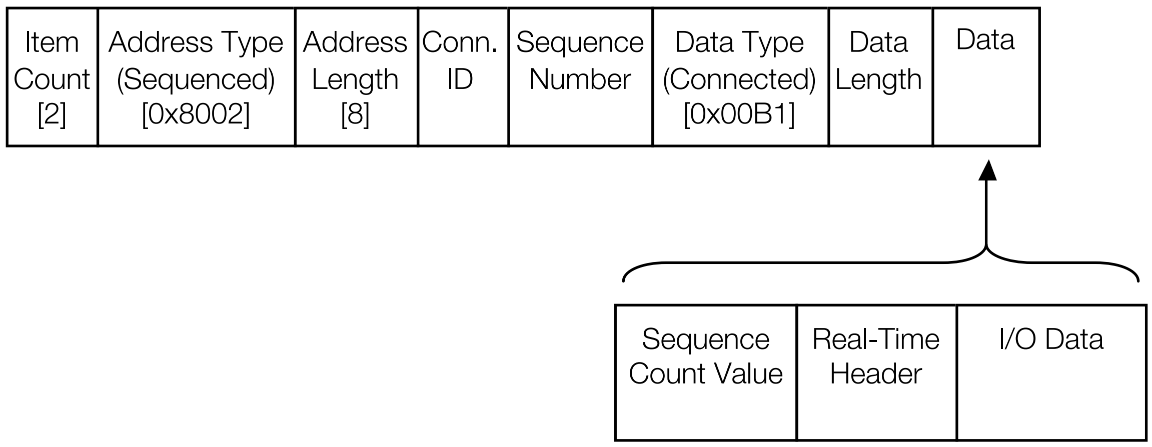


Figure 39 I/O Message Encapsulation

The data field contains the I/O data prefixed with a 16-bit sequence count value for the packet. I/O data transmission without the sequence count value is possible, but it is only used for CIP Safety connections. Run/Idle can be indicated within a Real-Time Header or by sending the packet with I/O data (Run) or without I/O data (Idle). The method used is indicated in the connection parameters of the Connection Manager section of the EDS. The Real-Time Header method is recommended [10] for use on EtherNet/IP for interoperability reasons and this is what is shown in Figure 39.

I/O messages from the originator to the target are typically sent as UDP unicast frames, while those sent from the target to the originator can be sent as UDP multicast or unicast frames. Multicast frames allow other EtherNet/IP devices to listen to the input data. To avoid having these UDP multicast frames propagating too widely over the network, the use of switches that support IGMP Snooping is highly recommended. IGMP (Internet Group Management Protocol [61]) is a protocol that allows the automatic creation of multicast groups. Using this functionality, the switch will automatically create and maintain a multicast group consisting of the devices that need to consume these multicast messages. Once the multicast groups have been established, the switch will direct such messages only to those devices that have subscribed to the multicast group of that message.

3.3.13. Connection Establishment

All connections on EtherNet/IP are established using a UCMM Forward_Open message (see chapter 2.3).

3.3.14. QuickConnect Connection Establishment

While most applications can wait several seconds until a connection is established, there are certain application scenarios that require a device to be operational with only a very short delay after the application of power. Typically, these are devices sitting on an exchangeable tool that a robot would pick up for certain manufacturing steps. In comparison to DeviceNet, EtherNet/IP devices are more complex and larger and more complex parts of the communication stack are typically implemented in software, so it will take more time to power up a device. The other additional complexity in EtherNet/IP is the TCP layer with a timeout behavior of its own. Furthermore, active infrastructure (switches) may take a long time to reboot. Under consideration of these conditions, the following method was developed for a fast establishment of I/O connections:

- If more than one EtherNet/IP device is mounted onto the exchangeable tool, embedded infrastructure with defined start-up behavior must be used;

- In preparation of a restart, every connection that participates in the QuickConnect application must be shut down using the Forward_Close service before disconnecting the device;

- When the device has responded to the Forward_Close request, it closes the TCP connection with the I/O Scanner;

- At the restart of any QuickConnect device, the I/O Scanner receives notification of power reapplication through a contact in the tool changer;

- The I/O Scanner then waits for a predetermined time (described in the EDS) before a connection is reestablished.

Using this methodology, startup times of less than 100 ms can be achieved with current technology; the first products are available on the market,

The full description of the QuickConnect functionality can be found in Appendix E of The EtherNet/IP Specification.

3.3.15. Device Classes

Four classes of device functionality are built with CIP. While they are not explicitly defined in the specification, they are useful for distinguishing among several classes of devices. The four classes are described here:

- The minimal device function is that of an explicit message server, which is used for explicit messaging applications only and acts as a target for Unconnected and (optionally) connected explicit messages, e.g., for program upload/download, data collection, status monitoring, etc.;

- The next device class is an I/O Server, which adds I/O messaging support to an explicit message server device and acts as a target for both Explicit and I/O messages, e.g., simple I/O Devices, Pneumatic Valves, AC Drives. These devices are also called I/O Adapters;

- Another device class is an explicit message client, which adds client support to explicit message server applications and acts as a target and as an originator for explicit messaging applications, e.g., computer interface cards, HMI devices;

- The most powerful type of device is an I/O Scanner, which adds I/O message origination support to the functionality of all the other device classes, and which acts as a target and as an originator for Explicit and I/O messages, e.g., PLCs, I/O Scanners.

3.3.16. Device Profiles

EtherNet/IP devices may utilize any of the device profiles described in the CIP Networks Library. As of the publication date of this book, no EtherNet/IP-specific device profiles have been defined.

3.3.17. Configuration

EtherNet/IP devices typically come with Electronic Data Sheets (EDS) as described in Section 2.7. For EDS-based configuration tools, the EDS should contain a Connection Manager section to describe the details of the connections that can be made into the device. This section basically mirrors what is contained in the Forward_Open message that a Connection Originator would send to the device. Multiple connections can be specified within an EDS that can then be chosen by the configuration tool.

An EDS also may contain individual parameters and/or a configuration assembly with a complete description of all parameters within this assembly. In many applications, the configuration assembly is transmitted as an attachment to the Forward_Open message.

3.3.18. Conformance Test

See section 6 of this publication for information on conformance testing.

3.3.19. Requirements for TCP/IP Support

In addition to the various requirements set forth in The EtherNet/IP Specification, all EtherNet/IP hosts are required to have a functional TCP/IP protocol suite and transport mechanism. The minimum host requirements for EtherNet/IP hosts are those covered in RFC 1122 [55], RFC 1123 [56], and RFC 1127 [57] 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 (Request for Comment) 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 The EtherNet/IP Specifications evolve.

All EtherNet/IP devices shall at a minimum support:

- Internet Protocol (IP version 4) (RFC 791 [48]);
- User Datagram Protocol (UDP) (RFC 768 [47]);
- Transmission Control Protocol (TCP) (RFC 793 [50]);
- Address Resolution Protocol (ARP) (RFC 826 [51]);
- Internet Control Messaging Protocol (ICMP) (RFC 792 [49]);
- Internet Group Management Protocol (IGMP) (RFC 1112 [54] & 2236 [61]);
- IEEE 802.3 (Ethernet) as defined in RFC 894 [52].

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 [58]);

- ARCNET (RFC 1201 [59]);

- FDDI (RFC 1103 [53]).

3.3.20. Coexistence of EtherNet/IP and Other Ethernet-Based Protocols

EtherNet/IP devices are encouraged, but not required, to support other Ethernet-based protocols and applications not specified in The EtherNet/IP Specification. For example, they may support HTTP, Telnet, FTP, etc. The EtherNet/IP protocol makes no requirements with regard to these protocols and applications.

Figure 40 shows the relationship between CIP and other typical Ethernet-based protocol stacks. Since EtherNet/IP, like many other popular protocols, is based on TCP/IP and UDP/IP, coexistence with many other services and protocols is not a problem, and CIP blends nicely into the set of already existing functions. This means that anyone who is already using some or all of these popular Ethernet services can add CIP without undue burden; the existing services like HTTP or FTP may remain as before, and CIP will become another service on the process layer.

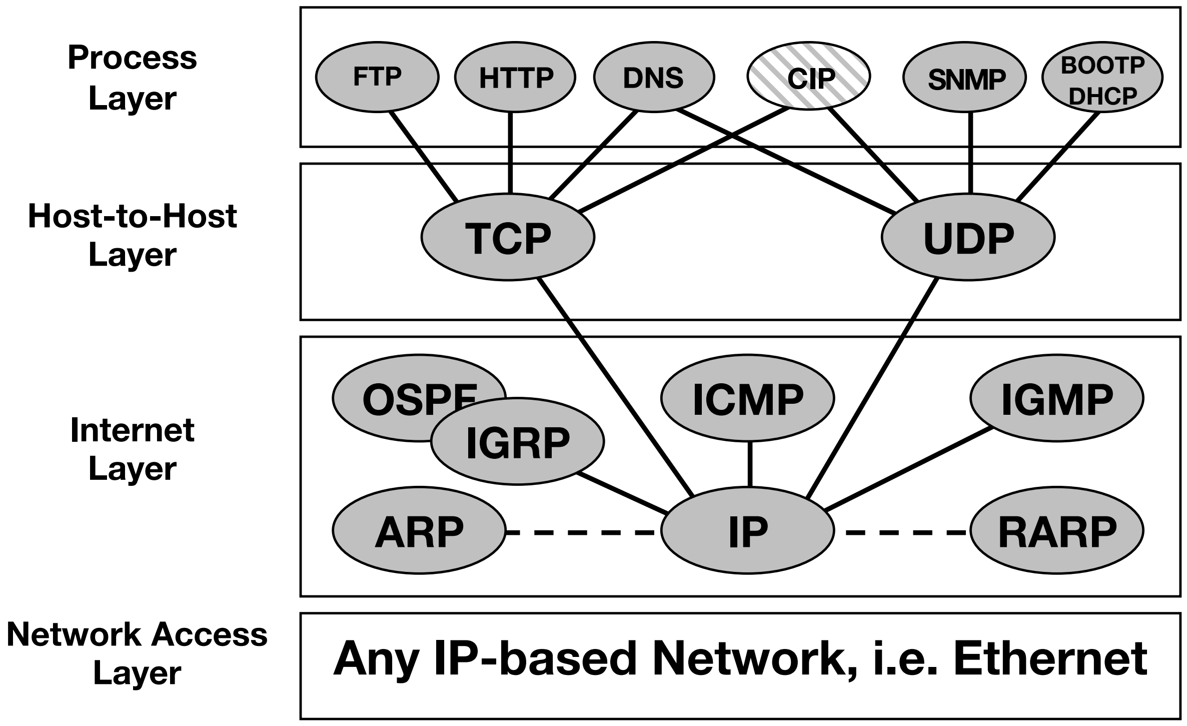


Figure 40 Relationship of CIP to Other Typical Ethernet Protocols

3.3.21. Ethernet Infrastructure

3.3.21.1. Traditional approach

To apply EtherNet/IP successfully to the automation world, the issue of determinism has to be considered. The inherent principle of the Ethernet bus access mechanism, whereby collisions are detected and nodes back off and try again later, cannot guarantee determinism. While Ethernet in its present form cannot be made strictly deterministic, there are ways to improve this situation.

First, the hubs typically used in many office environments must be replaced by more intelligent switches that will forward only those Ethernet frames intended for nodes connected to these switches. By using wire-speed switching fabric and full duplex switch technology, collisions are completely avoided; instead of colliding, multiple messages sent to the same node at the same time are queued up inside the switch and are then delivered one after another.

As already mentioned in Section 3.3.12.2, using switches that support IGMP Snooping is highly recommended.

If EtherNet/IP networks are to be connected to a general company network, this should always be done through a router. The router keeps the UDP multicast messages from propagating into the company network and ensures that the broadcast or multicast office traffic does not congest the control network. Even though the router separates the two worlds, it can be set up to allow the TCP/IP-based explicit messages to pass through so that a configuration tool sitting in a PC in the office environment may be capable of monitoring and configuring devices on the control network.

3.3.22. Devices with multiple Ethernet Ports

Chapter 6 of Volume 2 of the CIP Specifications describes a number of scenarios for devices with multiple Ethernet ports and how these scenarios are to be mapped in the object structure.

3.3.23. Ring and Linear Topologies

3.3.23.1. Linear Topology

Many end user applications benefit from connecting devices in a linear or ring topology. With such a topology, end devices typically have two Ethernet ports (with an embedded switch), and are connected in sequence, one device to the next.

With linear topology a failure of one node or a link between two nodes causes nodes on either side of the failure to be unreachable. By using a ring protocol implemented in the end devices, these devices may be configured in a ring topology so that a single-point failure does not prevent communication between the remainder of the functioning devices.

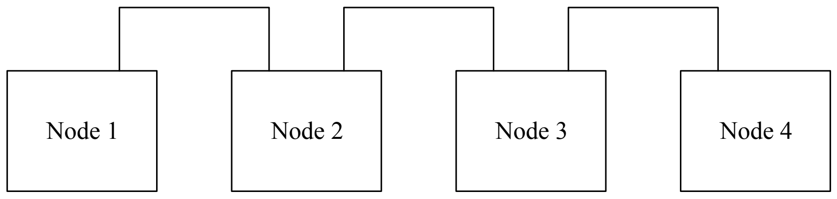


Figure 41 Linear Topology

3.3.23.2. Ring Topology with Device Level Ring (DLR)

3.3.23.2.1. DLR Overview

The EtherNet/IP specification includes the Device Level Ring (DLR) protocol, allowing multi-port devices to be connected in a ring topology. DLR provides for fast network fault detection and reconfiguration in order to support the most demanding control applications. For example, a ring network of 50 nodes implementing the DLR protocol has a worst case fault recovery time of less than 3ms.

The DLR protocol operates at Layer 2 (in the ISO OSI network model). The presence of the ring topology and the operation of the DLR protocol are transparent to higher layer protocols, such as TCP/IP and CIP, with the exception of a DLR object that provides a DLR configuration and diagnostic interface via CIP.

There are several classes of DLR implementation, as described below:

- **Ring Supervisor:**
  This class of devices is capable of being a ring supervisor. These devices must implement the required ring supervisor behaviors, including the ability to send and process Beacon frames at the default Beacon interval of 400 microseconds and may be user-configured for as fast as 100 microseconds Beacon interval. Ring supervisors must also send Announce frames, for those devices that rely on the Announce frame mechanism to detect a change in ring status;

- **Ring Node, Beacon-based:**
  This class of devices implements the DLR protocol, but without the ring supervisor capability. The device must be able to process and act on the Beacon frames sent by the ring supervisor. Beacon-based ring nodes must support Beacon rates from 100 microseconds to 100 milliseconds;

- **Ring Node, Announce-based:**
  This class of devices implements the DLR protocol, but without the ring supervisor capability. These devices do not have the capacity to process Beacon frames, so they simply forward Beacon frames received on one port to the other port and instead rely on Announce frames to indicate the ring state. Announce frames are sent at a much slower rate than Beacon frames.

3.3.23.2.2. Normal DLR Operation

A DLR network includes at least one node configured to be a ring supervisor, and any number of normal ring nodes. The ring supervisor sends special frames (“Beacon” and “Announce” frames) to detect ring fault and ring restoration.

Figure 42 illustrates the normal operation of a DLR network. Each node has two Ethernet ports, has implemented an embedded switch and supports DLR. When a ring node receives a packet on one of its Ethernet ports, it determines whether the packet needs to be received by the ring node itself (e.g., the packet has this node’s MAC address as the destination MAC address) or whether the packet should be sent out the other Ethernet port.

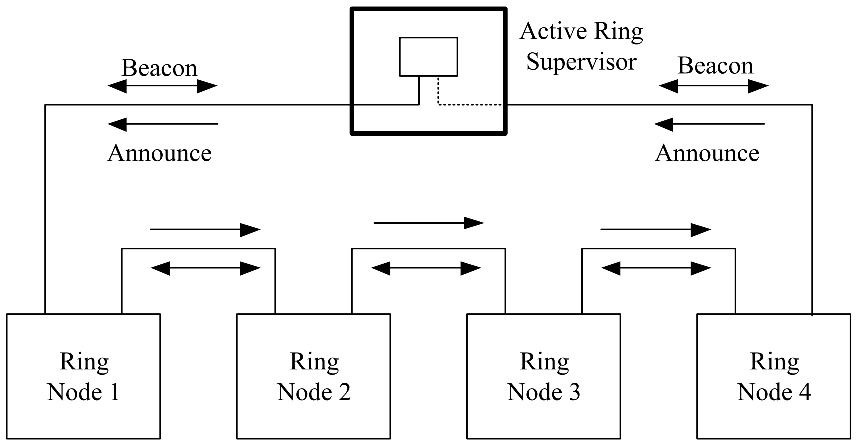


Figure 42 Normal DLR Operation

The active ring supervisor blocks traffic on one of its ports with the exception of few special frames and does not forward traffic from one port to other. This configuration avoids a network loop, so only one path exists between any two ring nodes during normal operation.

The active ring supervisor transmits a Beacon frame through both of its Ethernet ports once per beacon interval (400 microseconds by default). The active ring supervisor also sends Announce frames once per second. The Beacon and Announce frames serve several purposes:

- The presence of Beacon and Announce frames inform ring nodes to transition from linear topology mode to ring topology mode;

- Loss of Beacon frames at the supervisor enables detection of certain types of ring faults. (Note that normal ring nodes are also able to detect and announce ring faults);

- The Beacon frames carry a precedence value, allowing selection of an active supervisor when multiple ring supervisors are configured.

3.3.23.2.3. Ring Faults

Ring faults may include common link failures such as device power failure or media disconnection, or higher level failures where the physical layer is active but the device has failed.

The ring supervisor detects a ring fault directly via a Link Status message from a ring node, or indirectly via loss of Beacon frames. When a ring fault is detected, the active ring supervisor reconfigures the network by unblocking traffic on its previously blocked port and flushing its unicast MAC table. The supervisor immediately sends Beacon and Announce frames with the ring state value indicating that the ring is now faulted.

Ring nodes also flush their unicast MAC tables upon detecting loss of the Beacon in one direction, or upon receipt of Beacon or Announce frames with the ring state value indicating the ring fault state. Flushing the unicast MAC tables at both supervisor and ring nodes is necessary for network traffic to reach its intended destination after the network reconfiguration.

Figure 43 shows the network configuration after a link failure, with the active ring supervisor passing traffic through both of its ports.

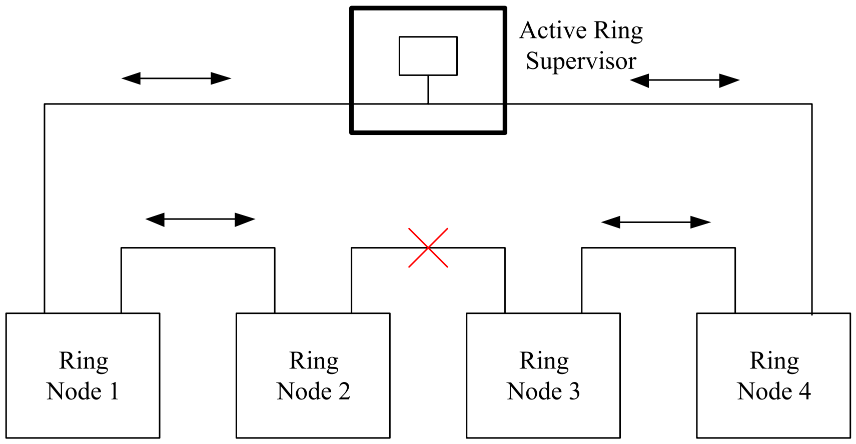


Figure 43 Network Reconfiguration after Link Failure

3.3.23.2.4. Further Information for Developers

Further details of the DLR protocol operation, including event tables, state diagrams, and other implementation requirements are included in Chapter 9 of the EtherNet/IP specification.

3.3.24. Use of the Rapid Spanning Tree Protocol (RSTP)

In addition to the DLR protocol, the Rapid Spanning Tree Protocol (RSTP) has also been allowed for use in conjunction with EtherNet/IP.

The Rapid Spanning Tree Protocol (RSTP) was originally designed for networks based on a tree topology where many devices are connected back to an Ethernet switch which in turn can be connected to other Ethernet switches. RSTP is a mature and widely accepted approach to solve the Ethernet ring recovery issue when one looks at the most current enhancements to the specification. The IEEE Standard 802.1D 2004 edition incorporated RSTP into that part of the standard. Changes were made by the IEEE Standards committee to RSTP that make it a suitable recovery mechanism for a ring topology for some automation applications.

Chapter 9 of The EtherNet/IP specification describes the use of RSTP within EtherNet/IP.

3.3.25. Tools

Tools for EtherNet/IP Networks can be divided into four groups:

- Physical layer tools are tools (hardware and/or software) that verify the integrity and conformance of the physical layer or monitor the quality of the data transmission;

- Commissioning tools: all EtherNet/IP devices need an IP address. In some cases, setting this address can be obtained only through the Ethernet network (see Section 3.3.9). In these cases, a BOOTP/DHCP server tool, such as the free BOOTP/DHCP routine downloadable from the Rockwell Automation website, is required;

- Configuration tools are software tools capable of communicating with individual devices for data monitoring and configuration purposes. Most configuration tools are EDS-based; however, more complex devices like I/O Scanners tend to have their own configuration applets that are only partially based on EDS files. Some of these tools support multiple access paths to the network, e.g., via suitable routing devices. High-level tools also actively query the devices on the network to identify them and monitor their health. Configuration tools also may be integrated into other packages like PLC programming software;

- Monitoring tools typically are PC-based software packages (e.g., traffic analyzers or "sniffers") that can capture and display the Ethernet frames on the network. A raw Ethernet frame display may be good enough in some instances, but using a tool that can display both raw Ethernet frames and provide multiple levels of frame interpretation (IP, TCP/UDP, EtherNet/IP header interpretation) is recommended. Due to the popularity of Ethernet, a large number of these tools are available, but not all of them support EtherNet/IP decoding.

In a typical installation, only a commissioning tool and a configuration tool are needed. Protocol monitoring tools are used mainly to investigate interoperability problems and to assist during the development process.

Turn to the ODVA Marketplace on the ODVA website to access a list of vendors that provide tools for EtherNet/IP.

3.3.26. Advice for Developers

Before any development of an EtherNet/IP product is started, the following issues should be considered in detail:

- What functionality (Device Classes, see Section 3.2.13) does the product require today and in future applications?
    - Explicit messaging server only;
    - I/O adapter functionality;
    - Explicit messaging client;
    - I/O scanner functionality.

- What are the physical layer requirements? Is IP 65/67 required or is IP 20 good enough?

- Will the development be based on commercially available hardware components and software packages (recommended) or designed from scratch (possible but costly)?

- What are the configuration requirements?

- What design and verification tools should be used?

- When and where will the product be tested for conformance and interoperability?
- What is an absolute must before products can be placed on the market (own the specification, have the company's own vendor ID, have the product conformance tested)?

Ethernet chipsets and associated base software packages are available from many vendors. For support of the EtherNet/IP part of development, refer to the ODVA website for a list of companies that can support EtherNet/IP development. An extended description of the development process is available for download from the ODVA website [14].

To help EtherNet/IP developers in creating their products, ODVA runs a series of so-called Implementor Roundtables for EtherNet/IP during which various aspects of EtherNet/IP are discussed. These workshops (with a North American and a European series) have created a number of documents with functionality recommendations for EtherNet/IP devices, see [9], [10], [11], [12] and [13]. EtherNet/IP devices are then tested against these recommendations during multi-vendor testing events, called Plugfests. Visit the ODVA web page to learn more about upcoming EtherNet/IP Implementor Roundtables and Plugfests.

3.3.27. EtherNet/IP Summary

Since its introduction in 2000, EtherNet/IP has shown remarkable growth in many applications that previously used traditional networks. This success (several million nodes installed to date) is largely attributed to the fact that EtherNet/IP uses standard, unmodified Ethernet to introduce real-time behavior into the Ethernet domain without sacrificing any of Ethernet's most useful features, such as company-wide access with standard and specialized tools through corporate networks.

A major strength of EtherNet/IP is the fact that it does not require a modified or highly segregated network: standard switches and routers used in the office world can be used for industrial applications without modification. At the same time, all existing transport-level or TCP/UDP/IP-level protocols can continue to be used without any need for special bridging devices. The substantially improved real-time behavior of CIP Sync and the introduction of CIP Safety also allow EtherNet/IP to be used in applications that currently require several dedicated networks.

Finally, as a member of the CIP family of networks, EtherNet/IP Networks can be combined into an overall CIP Network structure that allows seamless communication among CIP Networks, just as if they were only one network.

3.4. CompoNet

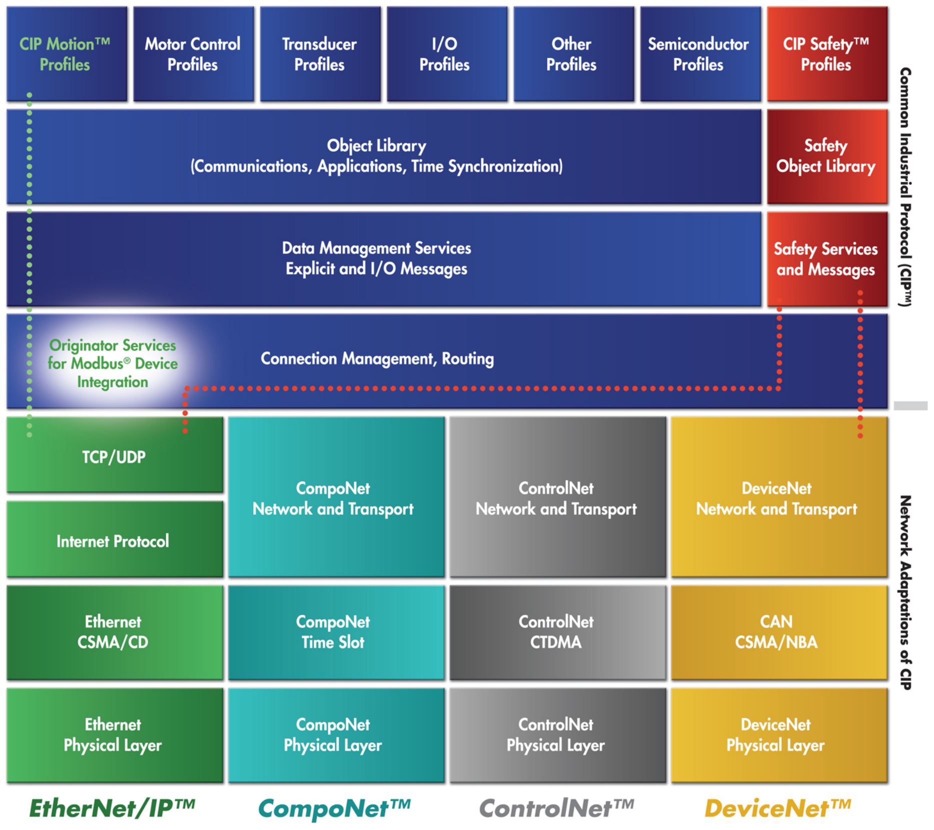
3.4.1. Introduction

CompoNet is a low-level network that provides high-speed communication between higher-level devices such as controllers and simple industrial devices such as sensors and actuators. IEC has published CompoNet as IEC 62026-7 in 2010 [26].

3.4.2. Relationship to Standards

Like other CIP Networks, CompoNet follows the Open Systems Interconnection (OSI) model, an ISO standard for network communications that is hierarchical in nature. Networks that follow this model define all necessary functions, from physical implementation up to the protocol and methodology to communicate control and information data within and across networks.

Figure 44 shows the relationship between CIP and CompoNet.



CIP Motion™ Profiles
Motor Control Profiles
Transducer Profiles
I/O Profiles
Other Profiles
Semiconductor Profiles
CIP Safety™ Profiles
Object Library (Communications, Applications, Time Synchronization)
Safety Object Library
Data Management Services Explicit and I/O Messages
Safety Services and Messages
Originator Services for Modbus® Device Integration
Connection Management, Routing
TCP/UDP
Internet Protocol
CompoNet Network and Transport
ControlNet Network and Transport
DeviceNet Network and Transport
Ethernet CSMA/CD
CompoNet Time Slot
ControlNet CTDMA
CAN CSMA/NBA
Ethernet Physical Layer
CompoNet Physical Layer
ControlNet Physical Layer
DeviceNet Physical Layer
EtherNet/IP™
CompoNet™
ControlNet™
DeviceNet™
Common Industrial Protocol (CIP™)
Network Adaptations of CIP

Figure 44 Relationship Between CIP and CompoNet

The CompoNet adaptation of CIP is described in Volume 6 of the CIP Networks Library [6]. All other features are based on CIP. CompoNet is also described in an international standard [26].

3.4.3. CompoNet Features

CompoNet supports both bit-level I/O slaves (BitIN and BitOUT slaves) and byte-level I/O slaves (WordIN and WordOUT slaves) simultaneously in one network. It also supports intelligent repeaters to expand the network flexibly and provide network diagnosis [37].

The main features of CompoNet are:

- Selectable data rates: 4 Mbps/3 Mbps/1.5 Mbps/93.75 kbps;
- Single master network with a large number of slave nodes: 384 slave devices maximum including WordIN: 64; WordOUT: 64; BitIN: 128; BitOUT: 128;
- Up to 64 repeaters per network to expand physical covering area and to adapt different cables;
- Up to 32 nodes (slaves and repeaters) per segment;

- I/O capacity: 1280 input/1280 output points;

- Support for flat 4-wire, round 4-wire and 2-wire cables in bus and branch topologies;

- Maximum of 3 segment layers. This means up to 2 repeaters are allowed between any slave and the master;

- 30/30/100/500 m maximum trunk cable distance with respect to data rates, 150/150/500/2500 m maximum distance between the most distant slaves with repeaters;

- Trunkline/dropline except for 4 Mbps;

- Efficient communication with multicast polling and Time Division Multiple Access (TDMA).

3.4.4. CompoNet Physical Layer

The physical layer of CompoNet has been designed specifically for this network; it does not reuse any existing open technology.

CompoNet uses a transformer-coupled transmission method and a Manchester-encoded signal on the wire; the principle circuit of the physical media attachment is shown in Figure 45.

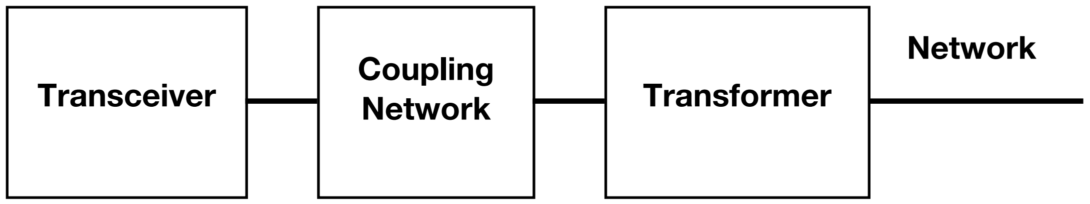


Figure 45 Physical Media Attachment of CompoNet

Master ports and slave ports use the same physical media attachment with only minor differences in the coupling network. To help vendors design their CompoNet physical media attachment, ODVA has published recommended circuits for both masters and slaves in Chapter 8 of The CompoNet Specification [6]. Vendors can also design their own circuits.

3.4.5. Frame Structure

A typical message frame is composed of the Preamble, Command Code, Command Code Dependent Block(s), and CRC, as shown in Figure 46.

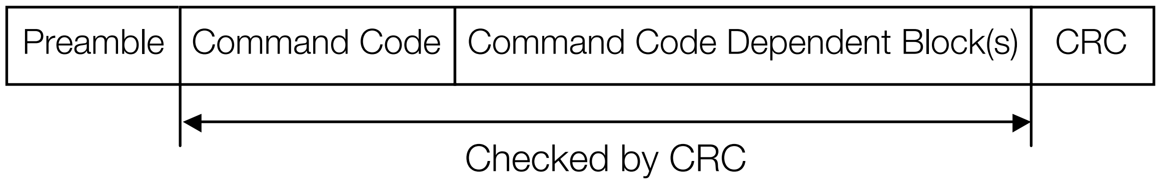


Figure 46 A General Frame

All frames use the same Preamble. Two types of CRC generator polynomials, CRC8 (8-bit) and CRC16 (16-bit), are used depending on which frame is used.

There are seven types of frames with varying lengths of command codes as shown in Figure 47:

Command Code							Meaning
B0	B1	B2	B3	B4	B5	B6	
0	0	0	1	x	x	x	OUT
0	0	1	1	x	x	x	TRG
0	1	x	x				CN
1	0						IN
1	1	1	x	x	x		A_EVENT
1	1	0	x	x	x		B_EVENT
0	0	0	0	1			BEACON

Figure 47 Command Code in Frames

- **OUT Frame:** This is a frame from the Master to the Slaves/Repeaters, which delivers OUT data to OUT slaves, specifies the group of slaves/repeaters that should report their status and synchronizes the slaves/repeaters to start time domain timers of the CN and IN frames. Data is organized in 16-bit words and transmitted, LSB first, word by word in ascending order;

- **TRG Frame:** This frame functions like the OUT frame except that it contains no output data; it is a trigger frame sent by the master in place of the OUT frame when it has no outputs to send;

- **CN Frame:** This frame is used by slaves/repeaters to report their connection status to the master and notify the master of a request to send an event;

- **IN Frame:** This is a frame from input slaves to the master with input data. Data is transmitted, LSB first, word by word in ascending order, if the size is greater than 16 bits;

- **A_EVENT Frame:** Any node on the network can send A_EVENT frames, used for acyclic message communications. Data is in 16-bit words and transmitted, LSB first, word by word in ascending order;

- **B_EVENT Frame** This frame is always originated by a master who sets data link parameters or reads the information as described in Chapter 3.4.9.2. Also, it is used to grant a slave/repeater permission to send an A_EVENT request or response;

- **BEACON Frame:** This frame specifies the data rate and sends the initial communication parameters to slaves/repeaters.

3.4.6. Protocol Adaptation

CompoNet uses connection-based I/O messaging and unconnected explicit messaging. Thus every device must have the UCMM (unconnected message manager) function.

3.4.7. Indicators and Switches

CompoNet does not require a product to have indicators. However, if a product includes indicators with any of the legends in Chapter 9 of The CompoNet Specification, they must follow the behavior specified in that chapter.

Chapter 9 of The CompoNet Specification describes how switches are used to set the MAC address and baud rate. The MAC address may also be set via explicit messaging.

3.4.8. Additional Objects

The CompoNet Specification defines two additional objects, the CompoNet link object and the CompoNet repeater object.

3.4.8.1. CompoNet Link Object (Class ID: 0xF7)

The CompoNet link object manages all aspects associated with the CompoNet link, in particular node address and baud rate as well as the switches associated with MAC address and baud rate. Furthermore, this object contains information on the allocation of the I/O communication. Within a master device, this object may also contain information on the slaves that are present on the network.

3.4.8.2. CompoNet Repeater Object (Class ID: 0xF8)

Every repeater device on CompoNet must support one instance of the CompoNet repeater object. Its main purpose is to monitor the power supply voltage of the subnet it connects to.

3.4.9. Network Access

3.4.9.1. Network Schedule

In a CompoNet network, the master controls bus communications according to its configuration. A master divides a communication cycle into several time domains or time slots.

CompoNet conducts arbitration under strict time supervision managed by the master. The communication cycle is partitioned into time domains as shown in Figure 48. Each node obtains the right to send data to the network within a specified time period after the completion of the OUT time domain.

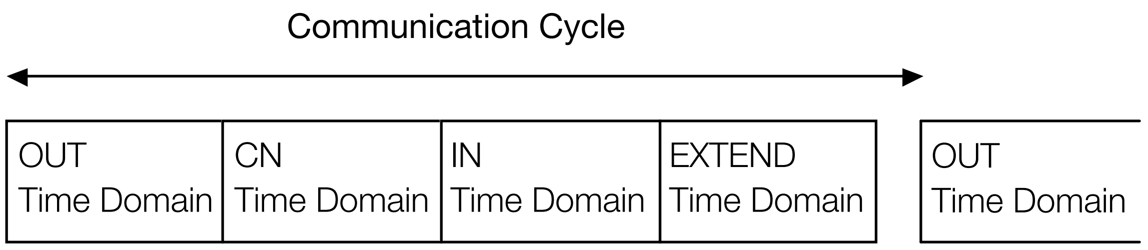


Figure 48 Time Domains

The first domain of each communication cycle is the OUT time domain. Subsequent domains are the CN time domain, the IN time domain, and the EXTEND time domain:

- **OUT Time Domain:** The master sends an OUT frame or a TRG frame in this period;

- **CN Time Domain:** CN frames are sent in this period. The number of CN frames sent in this time domain is determined by the master;

- **IN Time Domain:** IN frames are sent in this period consecutively by all input type devices;

- **EXTEND Time Domain:** The master executes message communications in this period. Event frames, i.e., A_EVENT frames and B_EVENT frames, can be sent in this period. BEACON frames shall be sent periodically. The master can send a BEACON before every OUT Time Domain starts or in an idle EXTEND Time Domain.

Figure 49 shows the sequence of frames in a Communication Cycle. The master starts the cycle by sending an OUT frame. The OUT frame is a broadcast message used to send output data to all OUT slaves. Each OUT slave consumes its output data (up to 16 bits) from its offset in the OUT frame. The completion of the OUT frame indicates the end of the OUT Time Domain and triggers slaves and repeaters to start the timers that allow them to correctly participate in the CN Time Domain and the IN Time Domain.

In the CN Time Domain, the slaves or repeaters addressed by the "CN Request MAC ID Mask" field in the OUT frame will transmit their CN frames at the pre-defined time sequence.

During the IN Time Domain IN frames are sent by any IN devices that are in the Participated State, at the pre-defined time sequence (see Figure 49). During the IN Time Domain, nodes in the EventOnly Substate do not transmit an IN frame.

During the EXTEND Time Domain, nodes may transmit an event command frame and possibly an immediate acknowledge frame, depending on the event type. These can be sent by the master, slaves or repeaters. The node designated in the event command frame’s Destination MAC ID field will send an event acknowledge frame, if required by the specific event command. CIP explicit messaging is done during this domain.

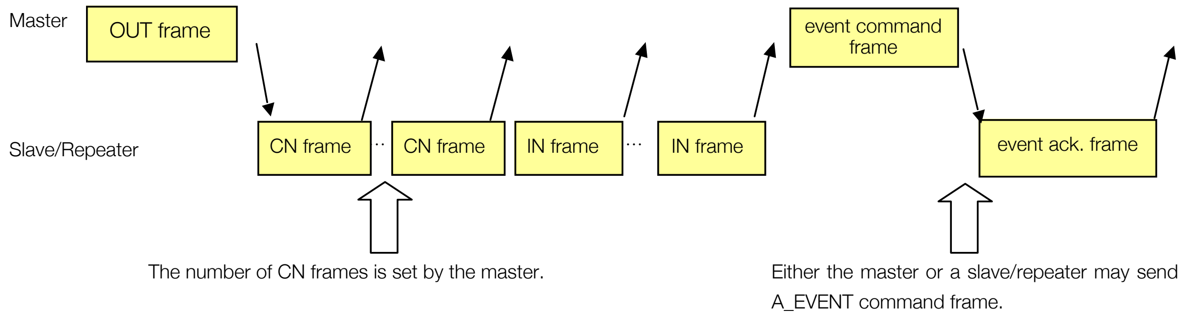

Master
OUT frame
event command frame
Slave/Repeater
CN frame
CN frame
IN frame
IN frame
event ack. frame
The number of CN frames is set by the master.
Either the master or a slave/repeater may send A_EVENT command frame.

Figure 49 A Typical Communication Cycle

3.4.9.2. Network Access

CompoNet has an algorithm that controls the network access of any of the slaves and repeaters. This is a combination of actions taken by the slave itself and commands from the master using STR (Status Read) and STW (Status Write) operation.

Starting from power-up, all devices first need to detect the data rate of the network. Once a device has detected the data rate from the BEACON frames sent by the master, it transitions to the “Non-participated” state. The device will stay in this state as long as it receives valid frames within the timeout period (650 ms for 93.75 Kbit/s, 200 ms for the other data rates) and no other event occurs. When the master has determined that it is ok for the slave to transition to the normal operating mode, i.e., no duplicate of the slave’s node number has been detected, it sends an STW_Run command to the slave and the slave then transitions to the “Participated” state, which is the normal operating state. The slave will leave this state and fall back to the “Non-participated” state either when it has experienced a network timeout (no OUT or TRG frames within the timeout period) or when it receives an STW_Standby command from the master.

The detection of duplicate node IDs is also a combination of master and slave action. The slave will go to the Communication Fault state if told to do so by the master through an STW_Dup command or when its CN Counter overflows due to communication errors caused by duplicate node IDs.

3.4.10. Explicit Messaging

CompoNet uses UCMM for explicit messaging; there is no connection-based explicit messaging. Explicit messages are encapsulated into A_EVENT frames as shown in Figure 50.

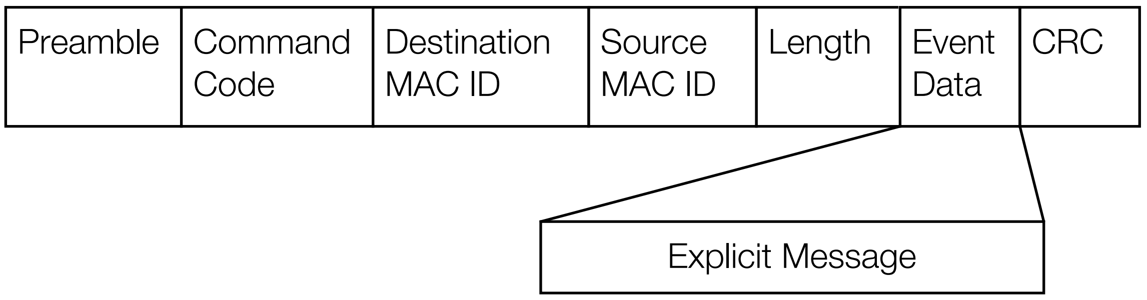


Figure 50 A_EVENT Frame Format

The format of an explicit message is defined to have 2 parts as shown in Figure 51 : Header and Service Data. The Header is the CompoNet-specific part containing information for flow control, fragmentation protocol and addresses in a word format. The Service Data part consists of the Request/Response Data as defined in the Message Router Request/Response format in Volume 1, Chapter 2 of the CIP Networks Library.

Word offset	B15	…	B00
0 to n	Header: Control Code, Destination MAC ID, Source MAC ID, SID/Extended SID, Size, Service Code, Class ID, Instance ID		
n+1 to 21	Service Data		

Figure 51 CompoNet Explicit Message Format

2 types of explicit message formats are defined:

- Compact - 1 Octet Class ID and Instance ID (required);

- Expanded – CIP EPATH (optional).

A fragmentation protocol is defined optionally for supporting long data transfers in explicit messages.

The SID/Extended SID field is used for reply matching. The client sets the value and the server echoes it back. Values used are specific to whether the master or the slave is the client in the transaction.

Explicit messaging transactions are subject to timeouts. The default timeout values can be changed by explicit messages.

3.4.11. I/O Messaging

I/O messages on CompoNet, like on any other CIP network, are always exchanged in a connection-based manner. Communication object s must be set up for this purpose. CompoNet uses a Predefined Allocation service to establish I/O communication between the master and the slaves. Since these I/O connections are not initially available when a device is powered on; they first have to be created. This is done by sending an Allocate Service to the CompoNet link object of the slave that is to be allocated. When an I/O connection is no longer needed, the slave(s) can be released by sending a Release Service to the slave's CompoNet link object.

CompoNet uses OUT frames to deliver output data to consuming slaves and to trigger IN frame transmission, IN frames to deliver produced data to the master and TRG frames to trigger IN frame transmission when the master has no output data to send.

OUT/TRG frames are monitored by I/O slaves to restart a watchdog timer. If an I/O slave has not received any OUT/TRG frames before the watchdog timer expires, it will produce a Timeout event. The timer value of the watchdog is 4 times the Expected Packet Rate (EPR) attribute in the I/O connection instance. A watchdog timeout event transitions the connection instance to the “Timeout” state and the application will be notified.

An OUT frame with I/O-Refresh disabled or a TRG frame indicates “Idle”, and an OUT frame with I/O-Refresh enabled indicates “Run”.

3.4.12. I/O Connection Establishment

As described in Section 3.4.11, I/O messages on CompoNet are always exchanged in a connection-based manner. Similar to DeviceNet, a CompoNet slave is allocated by the master by sending an Allocate Service to the CompoNet link object of the slave that is to be allocated.

When an I/O connection is no longer needed, a slave can be released by sending a Release Service to the slave’s CompoNet link object.

In contrast to DeviceNet, which also uses a predefined master/slave connection set, CompoNet only has one I/O data exchange mechanism (polled). Therefore, only one type of allocation may take place so there is only a single Allocation Choice bit defined in the Allocate service.

3.4.13. Device Profiles

3.4.13.1. Bit slave or Word slave

All existing profiles must be realized by using the minimum amount of CompoNet communication. The following rules apply to the adaptation of existing profiles:

- CompoNet frame rules must be observed;

- If an existing profile can be realized by a Bit Slave, it must be a Bit Slave;

- An existing profile must be realized with the minimal data length that is feasible.

3.4.13.2. Byte size differences

CompoNet uses the same definitions as other CIP network even though it has specific transmission frames. In order to agree with the data size in bytes as used by CIP, CompoNet needs some rules to align a CompoNet I/O frame (which counts I/O size in bits) with CIP objects (which typically count data length in bytes).

For bit-level slaves, the related I/O size shown in the connection object instance is rounded up to 1 byte. For example, A BitIN slave with 1 bit valid input data uses IN frames with 2 bits of data to deliver data on the wire, but the “produced_connection_size” (attribute 7) of its connection instance shall be “1”, which means 1 byte in CIP.

For byte-level slaves, the same sizes shall be used as with other CIP networks. If the CIP byte size does not match the CompoNet frame size, the size is rounded up to the next even number for the transmission. For example, a device with 3 bytes of application data will have size 3 in its application object (e.g., the assembly object) and size of 3 for the “produced_connection_size” (attribute 7) of its connection instance, but the IN frames on the wire have 32 bits of valid data.

3.4.14. Configuration

CompoNet devices typically come with Electronic Data Sheets (EDS) as described in Section 2.7.

To support EDS-based configuration, several CompoNet-specific EDS keywords have been added. With these new keywords and most CIP EDS keywords, CompoNet masters can be configured by a tool which can decode CompoNet EDS files. As a minimum (apart from the required sections), CompoNet EDSs should contain the CompoNet-specific sections that describe the I/O connections available in the slave.

An EDS also may contain individual parameters and/or a configuration assembly with a complete description of all parameters within this assembly.

CompoNet can also be configured by FDT/DTM, which is beyond the scope of CIP [46].

3.4.15. Advice for Developers

Before starting any CompoNet product development, the following issues should be considered in detail:

- What functionality does the product require today and in future applications?
    - Slave functionality;
    - Master functionality.

- What are the physical layer requirements? Is IP 65/67 required or is IP 20 good enough?

- What type of hardware should be chosen for this product?

- What kind of firmware should be used for this product? Will a commercially available communication stack be used?

- Will the development of hardware and/or software be done internally or will it be designed by an outside company?

- What are the configuration requirements?

- What design and verification tools should be used?

- What kind of configuration software should be used for this product? Will a commercially available software package be used, i.e., is an EDS adequate to describe the device or is custom software needed?

- When and where will the product be tested for conformance and interoperability?

- What is an absolute must before my products can be placed on the market (i.e., own the specification, have the company's own vendor ID, have the product conformance tested)?

A full discussion of these issues goes well beyond the scope of this publication, see [38], [39] instead. ODVA provides a developer's toolkit including working source code for CompoNet slaves and repeaters.

3.4.16. Conclusions

CompoNet is a well-adapted CIP network. It complies with the CIP object modeling, object addressing as well as the CIP communication model and its configuration rules. It is easy to realize CIP network routing and bridging. Combining with its advantages in aspects of data link and physical layer, CompoNet provides unique solutions with existing CIP resources to vendors and users.

All the changes in automation require consideration of the next generations of networking. In view of these trends and in view of technologies similar to those described above, it can be seen how much more successful a network can be if it is from the family of CIP Networks in all levels of hierarchy – instead of being patched together as a result of selecting isolated independent networks specific to each level.

4. Benefits of the CIP Family

CIP offers distinct benefits for two groups:

- Device manufacturers;
- Users of devices and systems.

4.1. Benefits for Device Manufacturers

For device manufacturers, a major benefit of using CIP is the fact that existing knowledge can be re-used from one protocol to another, resulting in lower training costs for development, sales and support personnel. Manufacturers also can reduce development costs, since certain parts (e.g., parameters, profiles) of the embedded firmware that are the same regardless of the network can be re-used from one network to the other. As long as these parts are written in a high-level language, the adaptation is simply a matter of running the right compiler for the new system.

Another important advantage for manufacturers is the easy routing of messages from one system to another. Any routing device can be designed very easily since there is no need to invent a "translation" from one system to another; both systems already speak the same language. Manufacturers also benefit from working with the same organizations for support and conformance testing.

4.2. Benefits for the Users of Devices and Systems

For users of devices and systems, a major benefit of using CIP is the fact that existing knowledge can be re-used from one protocol to another, e.g., through Device Profiles, and device behavior is identical from one system to another, resulting in lower training costs. Technical personnel and users do not have to make great changes to adapt an application from one type of CIP Network to another, and the system integrator can choose the CIP Network that is best suited to his application without having to sacrifice functionality.

Another important CIP advantage is the ease of bridging and routing between CIP Networks. Moving information among incompatible networks is always difficult and cumbersome since there is seldom a direct translation of all functionality on one network to another. This is where users can reap the full benefits of CIP. Forwarding data and messages from top to bottom and back again is very easy to implement and uses very little system overhead. There is no need to translate from one data structure to another – they are the same. Services and status codes share the same benefit, as these, too, are identical over all CIP Networks. Finally, creating a message that runs through multiple hops of CIP Networks is simply a matter of inserting the full path from the originating to the target device. Not a single line of code or any other configuration is required in the routing devices, resulting in fast and efficient services that are easy to create and maintain. Even though these networks may be used in different parts of the application, messaging from beginning to end really functions as if there is only one network.

Finally, the producer/consumer mechanisms used in all CIP Networks provide highly efficient use of transmission bandwidth, resulting in system performance that often is much higher than that of other networks running at higher raw baud rates. With CIP, only the truly important data is transmitted, rather than old data being repeated over and over again.

Planned and future protocol extension will always be integrated in a manner that allows coexistence of "normal" devices with "enhanced" devices like those supporting CIP Sync and/or CIP Safety. Therefore, no strict segmentation into "Standard", CIP Sync and CIP Safety networks is required unless there is a compelling reason, e.g., unacceptably high response times due to high bus load.

5. Application Layer Enhancements

5.1. CIP Sync and CIP Motion

5.1.1. General Considerations

While CIP Networks [1]-[4], [6] provide real-time behavior that is appropriate for many applications, a growing number of applications require even tighter control of certain real-time parameters. Let us have a look at some of these parameters:

- **Real-Time:**

This term is being used with a variety of meanings in various contexts. For further use in this section the following definition is used: A system exhibits real-time behavior when it can react to an external stimulus within a predetermined time. How short or how long this time is depends on the application. Demanding industrial control applications require reactions in the millisecond range while, in some process control applications, a reaction time of several seconds or more is sufficient;

- **Determinism:**

A deterministic system allows for a worst-case scenario (not a prediction or a probability) when deciding on the timing of a specific action. Industrial communication systems may offer determinism to a greater or lesser degree, depending on how they are implemented and used. Networks featuring message transmission at a predetermined point in time, such as ControlNet, Sercos Interface and Interbus-S, are often said to offer absolute determinism. On the other hand, networks such as Ethernet may become non-deterministic under certain load conditions; specifically, when it is deployed in half-duplex mode with hubs. However, when Ethernet is deployed with full-duplex, high-speed switches, it operates in a highly deterministic manner (see Section 3.3.21);

- **Reaction Time:**

In an industrial control system, the overall system reaction time is what determines real-time behavior. The communication system is only one of several factors contributing to the overall reaction time. In general, reaction time is the time from an input stimulus to a related output action;

- **Jitter:**

This term defines the time deviation of a certain event from its average occurrence. Some communication systems rely on very little message jitter, while most applications require only that a certain jitter is not exceeded for actions at the borders of the system, such as input sampling jitter and output action jitter;

- **Synchronicity:**

Distributed systems often require certain actions to occur in a coordinated fashion, i.e., the actions must take place at a predetermined moment in time, independent of where the action is to take place. A typical application is coordinated motion or electronic gearing. Some of these applications require synchronicity in the microsecond range;

This is a system's capability to process a certain amount of data within a specific time span. In communication systems, protocol efficiency, the communication model (e.g., producer/consumer) and end point processing power are most important, while the wire speed only sets the limit of how much raw data can be transmitted across the physical media. CIP Sync is a communication principle that enables synchronous low jitter system reactions without the need for low jitter data transmission. This is of great importance in systems that do not provide absolute deterministic data transmission or where it is desirable for a variety of higher layer protocols to run in parallel with the application system protocol. The latter situation is characteristic of Ethernet. Most users of TCP/IP-based Ethernet want to keep using it as before without the need to resort to a highly segregated network segment to run the real-time protocol. The CIP Sync communication principle meets these requirements.

5.1.2. Using IEEE 1588 Clock Synchronization

The published IEEE standard 1588 – Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems [40] – lays the foundation for a precise synchronization of real-time clocks in a distributed system.

An IEEE 1588 system consists of a Time Master that distributes its system time to Time Slaves in a tree-like structure. The Time Master may be synchronized with another real-time clock further up in the hierarchy while the Time Slaves may be Time Masters for other devices "below" them. A Time Slave that is Time Master to another set of devices (typically, in another part of the system) is also called a Boundary Clock. The time distribution is done by multicasting a message with the actual time of the master clock. This message originates in a relatively high layer of the communication stack and, therefore, the actual transmission takes place at a slightly later point in time. Also, the stack processing time varies from one message to another. To compensate for this delay and its jitter, the actual transmission time can be captured in a lower layer of the communication stack, such as noting the "transmit complete" feedback from the communication chip. This update time capture is then distributed in a follow-up message. The average transmission delay also is determined so that the time offset between the master and slave clock can be compensated.

This protocol has been fully defined for Ethernet UDP/IP systems, and the protocol details for further industrial communication systems will follow. The clock synchronization accuracy that can be achieved with this system depends largely on the precision time capture of the master clock broadcast message. Hardware-assisted time capture systems can reach a synchronization accuracy of 250 ns or less. Some Ethernet chip manufacturers offer integrated IEEE 1588 hardware support.

5.1.3. Additional Object

CIP Sync requires the addition of a time synchronization object.

5.1.3.1. Time Sync Object, (Class ID: 0x43)

The time sync object manages the real-time clock inside a CIP Sync device and provides access to the IEEE 1588 timing information.

5.1.4. Fundamentals of CIP Sync Communication

Real-time clocks coordinated through the IEEE 1588 protocol do not, of their own accord, constitute a real-time system yet. Additional details showing how time stamping is used for input sampling and for the coordination of output actions need to be added. Some Device Profiles need to be extended as well to incorporate time information in their I/O assemblies.

5.1.5. Message Prioritization

Combining these three elements (Sections 5.1.2, 5.1.3 and 5.1.4) in conjunction with a collision-free infrastructure (see Section 3.3.21) is sufficient to build a real-time system. However, it is necessary to consider all traffic within the system and to arrange all application messages containing time-critical data in such a way that they are guaranteed to arrive at all consumers in time. When other Ethernet protocols – such as HTTP or FTP, which may have very long frames – need to coexist in the same system, careful configuration may be required. Ethernet frames with up to 1,500 bytes of payload (approximately 122 µs long in a 100-Mbit/second system) can easily congest the system and delay important messages by an undetermined amount of time, possibly too long for the system to function correctly.

This is where message prioritization, known as Quality of Service (QoS) in the Ethernet world, becomes an important element. EtherNet/IP defines common usage for two standard QoS mechanisms: Differentiated Services (Diffserv) and IEEE 802.1Q tagged frames. These schemes are supported by many switches available today. QoS allows preferential treatment of Ethernet frames in such a way that those frames with the highest priority will jump the message queues in a switch and will be transmitted first. Messages with high priority will be transmitted while those with lower priority typically have to wait. Suitable priority assignments for all time-critical messages then guarantee their preferential treatment. Standard EtherNet/IP and other Ethernet messages will receive low or no priority and thus have to wait until all higher-priority messages have passed. Once this prioritization scheme is implemented, one full-length frame can be accommodated within each communication cycle consisting of a set of prioritized input (port A through port E) and output (port F) messages. Figure 52 illustrates this process.

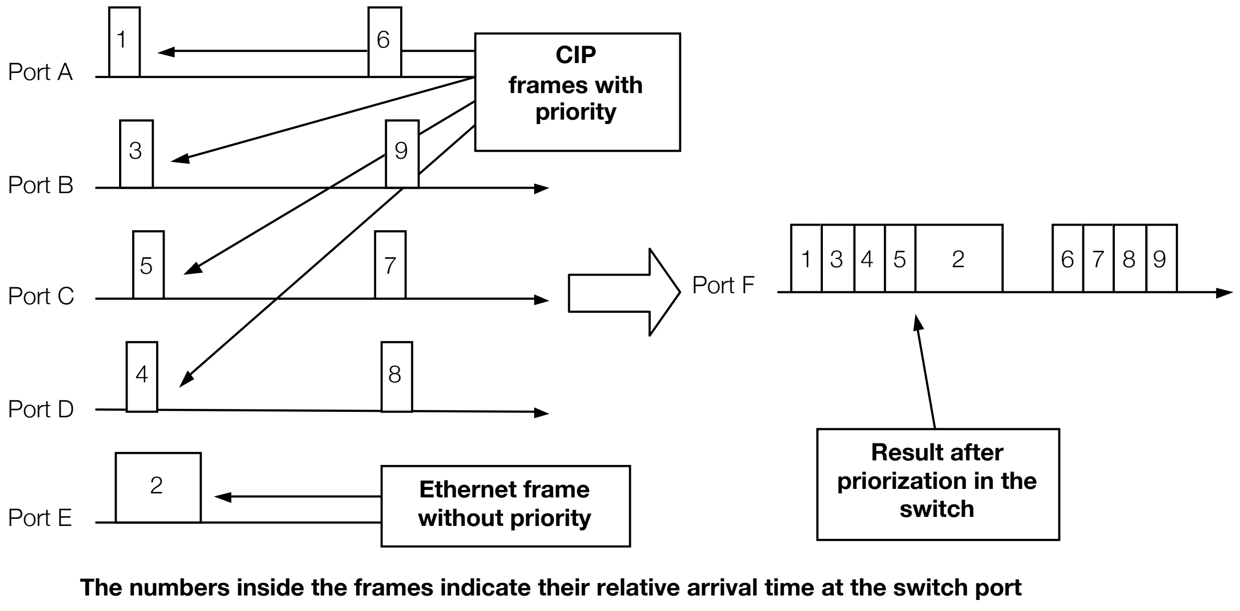

Port A
1
6
CIP frames with priority
Port B
3
9
Port C
5
7
Port D
4
8
Port E
2
Ethernet frame without priority
Port F
1 3 4 5 2
6 7 8 9
Result after priorization in the switch
The numbers inside the frames indicate their relative arrival time at the switch port

Figure 52 Ethernet Frame Prioritization

The overall approach to QoS for EtherNet/IP calls for devices to mark their packets with a priority value, using Diffserv Code Points and/or 802.1D priority values. By explicitly marking packets with a priority value, switches and routers are able to differentiate EtherNet/IP traffic from non-critical traffic, as well as differentiate specific EtherNet/IP traffic streams (e.g., IEEE 1588 vs. I/O vs. explicit messaging).

The following list summarizes the QoS behavior for EtherNet/IP:

- For CIP transport class 0 and 1 connections (i.e., UDP-based), there is a defined mapping of CIP priorities to 802.1D priorities and DiffServ Code Points;
- For UCMM and CIP transport class 3 connections (i.e., TCP-based), there is a single defined DiffServ Code Point and 802.1D priority value;
- For PTP (IEEE 1588) messages, there are DiffServ Code Points and 802.1D priority values corresponding to the two different types of PTP messages;
- When QoS is implemented, the default behavior is to mark packets with DSCP values. Devices may optionally support sending and receiving 801.1Q frames with the corresponding priority values. If supported, sending tagged frames is disabled by default in order to prevent device interoperability problems. The end user is responsible for enabling the tagged frame behavior and ensuring interoperability between devices;
- The QoS Object provides a means to configure DSCP values, and a means to enable/disable sending of 802.1Q tagged frames;
- There are no requirements for devices to mark traffic other than CIP or IEEE 1588, but devices are free to do so.

5.1.6. Applications of CIP Sync

Typical applications for CIP Sync are time-stamping sensor inputs, distributed time-triggered outputs and distributed motion, such as electronic gearing or camming applications. For example, in motion applications, sensors sample their actual positions at a predetermined time, i.e., in a highly synchronized way, and transmit them to the application master that coordinates the motion. The application master then calculates the new reference values and sends them to the motion drives. Using CIP Sync, the communication system is not required to have extremely low jitter; it is sufficient to transmit all time-critical messages, and their exact arrival time becomes irrelevant. The assignment of suitable priorities to CIP Sync communication guarantees that all time-critical messages always have the bandwidth they need, and all other traffic automatically is limited to the remaining bandwidth.

As a result of these measures, CIP Sync devices can coexist side by side with other EtherNet/IP devices without any need for network segmentation or special hardware. Even non-EtherNet/IP devices – provided they do not override any of the CIP Sync prioritizations – can be connected without any loss of performance in the CIP Sync application.

5.1.7. Expected Performance of CIP Sync Systems

As mentioned, CIP Sync systems can be built to maintain a synchronization accuracy of better than 250 ns, in many cases without the use of Boundary Clocks. The communication cycle and thus the reaction delay to unexpected events is largely governed by the number of CIP Sync devices in a system. Allowing some bandwidth (approx. 40%) for non-CIP Sync messages as described in Section 5.1.5, the theoretical limit (close to 100% wire load) for the communication cycle of a CIP Sync system based on a 100-Mbit/s Ethernet network is around 500 µs for 30 coordinated motion axes, with 32 bytes of data each.

5.1.8. CIP Sync Summary

CIP Sync is a natural extension of the EtherNet/IP system into the real-time domain. Unlike many other proposed or existing real-time extensions to other protocols, CIP Sync does not require any strict network segmentation between high-performance, real-time sections and other parts of the communication system. CIP Sync provides the ability to mix parallel TCP/IP-based protocols with industrial communication architectures of any size without compromising performance.

CIP Sync currently has been applied to EtherNet/IP, and an extension to other CIP implementations will follow.

5.1.9. CIP Sync and CIP Motion

CIP Motion utilizes CIP Sync to manage real-time motion control. As discussed previously, CIP Sync utilizes the IEEE-1588 "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" [40] to synchronize devices to a very high degree of accuracy. CIP Sync encapsulates the IEEE-1588 services which measure network transmission latencies and correct for infrastructure delays. The result is the ability to synchronize distributed clocks to within hundreds of nanoseconds of accuracy, or less.

Once all the devices in a control system share a synchronized, common, understanding of system time, real-time control can be accomplished by including time as a part of the motion information. Unlike the traditional approaches to motion control, the CIP Motion solution doesn't schedule the network to create determinism. Instead, CIP Motion delivers the data and the timestamp for execution as a part of the packet on the network. This allows motion devices to plan and follow positioning path information according to a pre-determined execution plan. Since the motion controller and the drives share a common understanding of time, the motion controller can tell the drive where to go – and what time to be there. This direct use of time in the data packet frees the network from the constraint of a rigid data delivery schedule. If data delivery fluctuates slightly on the network – the motion execution is unaffected.

There are many benefits to this approach. Since the network is not "scheduled" there is flexibility in the amount of data that can be sent back from each device. During runtime, a drive can be re-configured to send more or less data depending on the needs of the application. In addition, devices can be added or removed from the system because specific time slots are not allocated from the network bandwidth. The motion data packets that move between drives and controllers contain all the relevant information required for real-time motion execution; as long as basic clock synchronization is maintained - time is used as the event for execution – not the data delivery itself.

5.1.10. CIP Motion

CIP Motion was added to the CIP Networks Library in 2006, with the addition of the CIP Motion Axis Object and the CIP Motion Device Profile.

5.1.10.1. CIP Motion Profile

The CIP application profile used on EtherNet/IP provides a comprehensive set of services and device profiles that provide a wide range of functionality and device support. CIP Motion extends the CIP capability by defining extensions focused on drive control as listed below:

- Torque, velocity, or position control of servo and VFDs (Variable Frequency Drives)

- Servo drive and VFD configuration, status, and diagnostic parameters

- Support for feedback-only axes that can provide reference information for camming and lineshafting applications;

- Unicast control-to-drive communications;

- Multicast peer-to-peer communications (future).

The CIP Motion profile is designed to minimize the differences between servo drive and VFD handling. This facilitates features like common configuration services, common status and diagnostic services, and common application instruction support for servo drives and VFDs making them interchangeable at the application level.

The CIP Motion profile takes advantage of the latest advances in motion control technology to provide a comprehensive, state of the art profile. Extensive use of floating point data eliminates the complexity typically associated with integer math and scaling. The profile focuses on a simplified slave interface, making it easier for drive vendors to develop products that connect to the EtherNet/IP network and utilize the CIP Motion extensions.

5.2. CIP Safety

Like other safety protocols based on industry standard networks, CIP Safety adds additional services to transport data with high integrity. Unlike other networks, CIP Safety presents a scalable, network-independent approach to safety network design, one in which the safety services are described in a well-defined layer. Since safety functionality is incorporated into each device – rather than in the network infrastructure – CIP Safety allows both standard and safety devices to operate on the same open network. This capability gives users a choice of network architectures – with or without a safety PLC – for their functional safety networks. This approach also enables safety devices from multiple vendors to communicate seamlessly across standard CIP Networks to other safety devices without requiring difficult-to-manage gateways.

A complete definition of all details of CIP Safety can be found in Volume 5 [5].

5.2.1. General Considerations

Hardwired safety systems employ safety relays that are interconnected to provide a safety function. Hardwired systems are difficult to develop and maintain for all but the most basic applications. Furthermore, these systems place significant restrictions on the distance between devices.

Because of these issues, as well as distance and cost considerations, implementing safety services on standard communication networks is highly preferable. The key to developing safety networks is not to create a network that cannot fail, but to create a system where failures in the network cause safety devices to go to a known state. If the user knows which state the system will go to in the event of a failure, they can make their application safe. But this means that significantly more checking and redundant coding information is required.

So, to determine the additional safety requirements, the German Safety Bus committee [32] implemented and later extended an existing railway standard [33]. This committee provided design guidelines to safety network developers to allow their networks and safety devices to be certified according to IEC 61508 [24]. The latest version of this document has been published as GS-ET-26 [34].

Based on these standards, CIP was extended for high-integrity safety services. The result is a scalable, routable, network-independent safety layer that alleviates the need for dedicated safety gateways. Since all safety devices execute the same protocol, independent of the media on which they reside, the user approach is consistent and independent of media or network used.

CIP Safety is an extension to standard CIP that has been approved by TÜV Rheinland for use in IEC 61508 SIL 3 and EN 954-1 Category 4 applications, now ISO 13849-1, performance level e [35]. It extends the model by adding CIP Safety application layer functionality, as shown in Figure 53. The additions include several safety-related objects and Safety Device Profiles with specific implementation details of CIP Safety as implemented on DeviceNet, EtherNet/IP and Sercos.

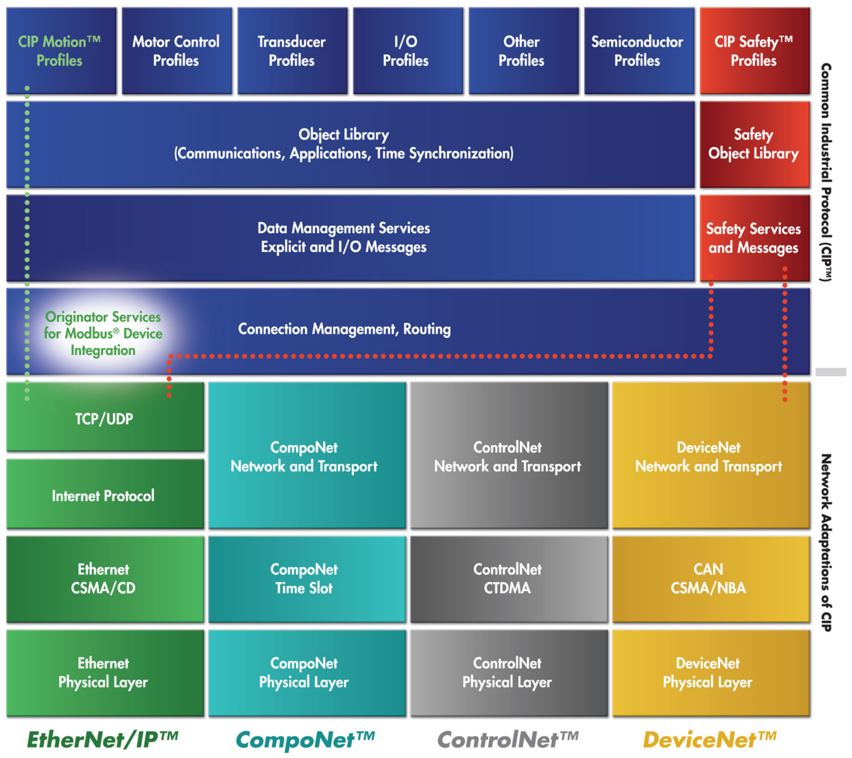

CIP Motion™ Profiles
Motor Control Profiles
Transducer Profiles
I/O Profiles
Other Profiles
Semiconductor Profiles
CIP Safety™ Profiles
Object Library (Communications, Applications, Time Synchronization)
Safety Object Library
Data Management Services Explicit and I/O Messages
Safety Services and Messages
Originator Services for Modbus® Device Integration
Connection Management, Routing
Common Industrial Protocol (CIP™)
TCP/UDP
Internet Protocol
CompoNet Network and Transport
ControlNet Network and Transport
DeviceNet Network and Transport
Ethernet CSMA/CD
CompoNet Time Slot
ControlNet CTDMA
CAN CSMA/NBA
Ethernet Physical Layer
CompoNet Physical Layer
ControlNet Physical Layer
DeviceNet Physical Layer
Network Adaptations of CIP
EtherNet/IP™
CompoNet™
ControlNet™
DeviceNet™

Figure 53 CIP Communication Layers Including Safety

Because the safety application layer extensions do not rely on the integrity (see Section 5.2.3) of the underlying standard CIP as described in Section 2 and data link layers as described in Sections 3.1, 3.2 and 3.3, single channel (non-redundant) hardware can be used for the data link communication interface. This same partitioning of functionality allows the use of standard routers for safety data, as shown in Figure 54. Routing safety messages is possible because the end device is responsible for ensuring the integrity of the data. If an error occurs during data transmission or in the intermediate router, the end device will detect the failure and take appropriate action.
This routing capability allows the creation of safety cells on one network, e.g., DeviceNet, with quick reaction times to be interconnected with other cells via a backbone network such as EtherNet/IP for interlocking, as shown in Figure 55. Only the safety data that is needed is routed to the required cell, which reduces individual bandwidth requirements. The combination of rapidly responding local safety cells and the inter-cell routing of safety data allows users to create large safety applications with fast response times. Another benefit of this configuration is the ability to multicast safety messages across multiple networks.

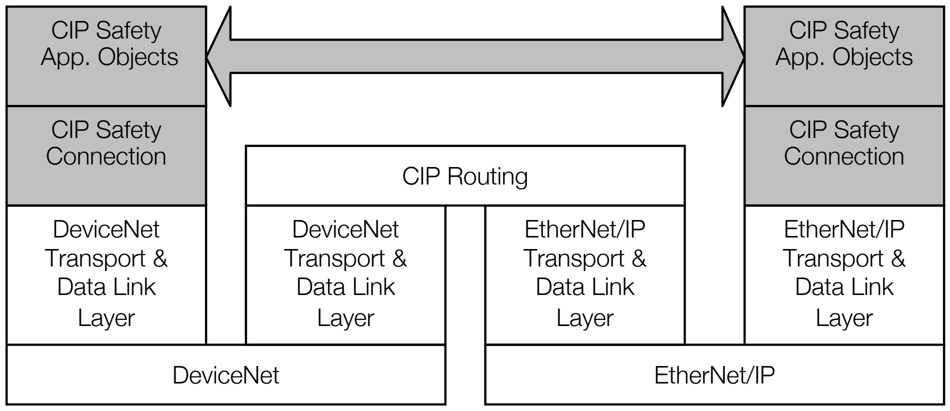

CIP Safety App. Objects
CIP Safety Connection
DeviceNet Transport & Data Link Layer
CIP Routing
DeviceNet Transport & Data Link Layer
EtherNet/IP Transport & Data Link Layer
CIP Safety App. Objects
CIP Safety Connection
EtherNet/IP Transport & Data Link Layer
DeviceNet
EtherNet/IP

Figure 54 Routing of Safety Data

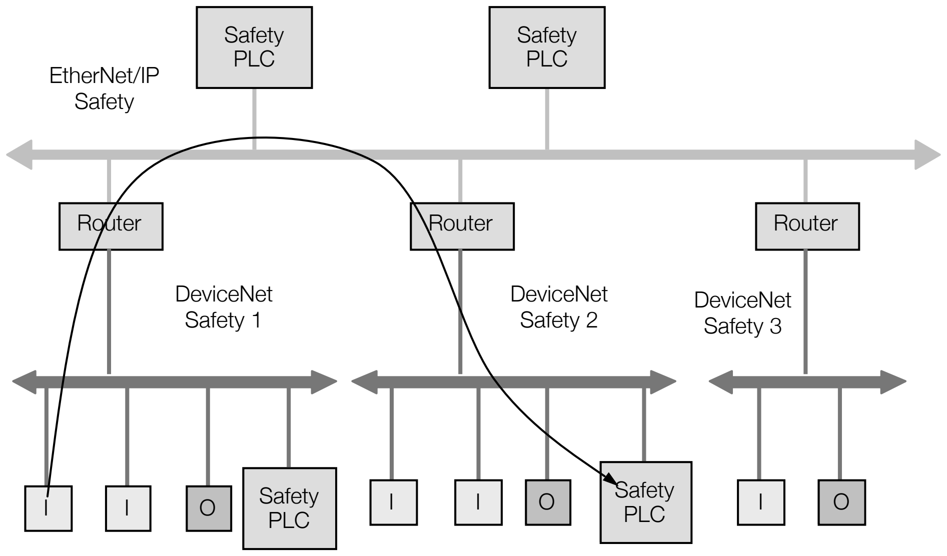


Figure 55 Network Routing

5.2.2. Implementation of Safety

As indicated in Figure 53, all CIP Safety devices also have underlying standard CIP functionality. The extension to the CIP Safety application layer is specified using a Safety validator object. This object is responsible for managing the CIP Safety Connections (standard CIP Connections are managed through communication objects) and serves as the interface between the safety application objects and the link layer connections, as shown in Figure 56. The safety validator ensures the integrity of the safety data transfers by applying the measures described in Section 5.2.3.

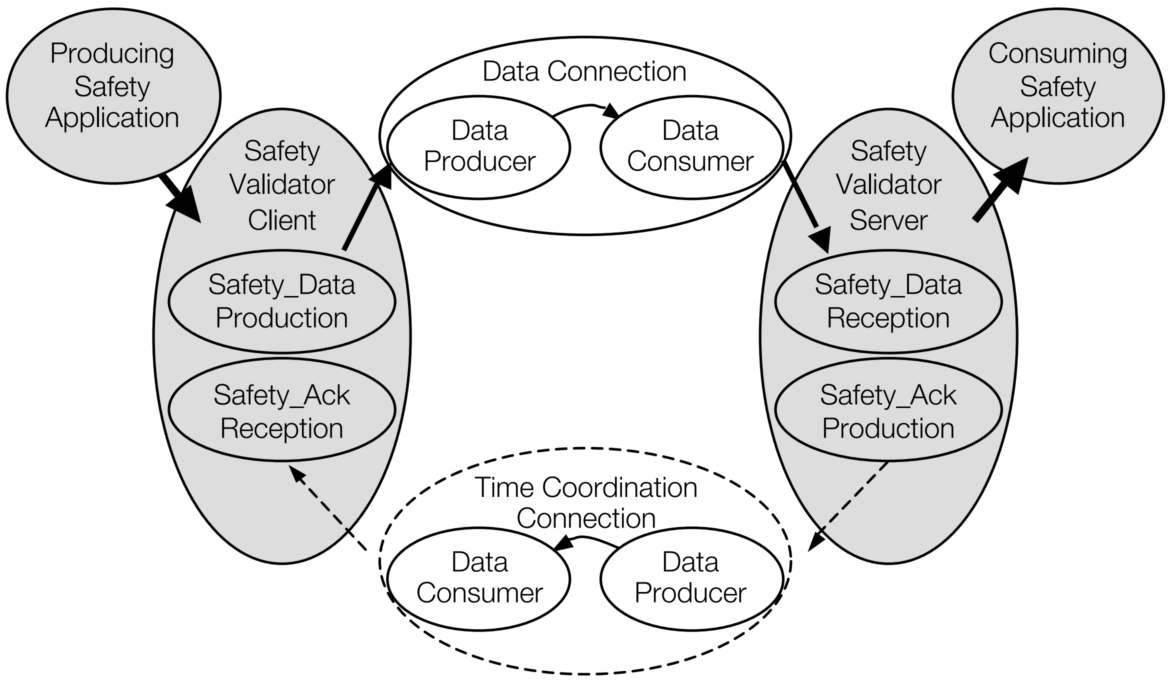


Figure 56 Relationship of Safety Validators

Functions performed by the safety validator object:

- The producing safety application uses an instance of a Client Validator to produce safety data and ensure time coordination;

- The client uses a link data producer to transmit the data and a link consumer to receive time coordination messages;

- The consuming safety application uses a Server Validator to receive and check data;

- The server uses a link consumer to receive data and a link producer to transmit time coordination messages.

- The link producers and consumers have no knowledge of the safety packet and fulfill no safety function

The responsibility for high-integrity transfer and checking of safety data lies within the safety validators.

5.2.3. Ensuring Integrity

CIP Safety does not prevent communication errors from occurring; rather, it ensures transmission integrity by detecting errors and allowing devices to take appropriate actions. The safety validator is responsible for detecting these communication errors. The nine communication errors which must be detected are shown in Figure 57, along with the five measures CIP Safety uses to detect these errors [33].

	Measures to detect communication errors				
Communication Errors	Time Expectation via time stamp	ID for send and receive	Safety CRC	Redundancy with Cross Checking	Diverse measure
Message Repetition	X		X		
Message Loss	X		X		
Message Insertion	X	X	X		
Incorrect Sequence	X		X		
Message Corruption			X	X	
Message Delay	X				
Coupling of safety and safety data		X			
Coupling of safety and standard data	X	X	X	X	X
Increased age of data in bridge	X				

* The Safety CRC provides additional protection for communication errors in fragmented messages.

Figure 57 Error Detection Measures

5.2.3.1. Time Expectation via Time Stamp

All CIP Safety data are produced with a time stamp that allows Safety Consumers to determine the age of the produced data. This detection measure is superior to the more conventional reception timers. Reception timers can tell how much time has elapsed since a message was last received, but they do not convey any information about the actual age of the data. A time stamp allows transmission, media access/arbitration, queuing, retry and routing delays to be detected.

Time is coordinated between producers and consumers using ping requests and ping responses, as shown in Figure 58. After a connection is established, the producer generates a ping request, which causes the consumer to respond with its consumer time. The producer will note the time difference between the ping production and the ping response and store this as an offset value. The producer will add this offset value to its producer time for all subsequent data transmissions. This value is transmitted as the time stamp. When the consumer receives a data message, it subtracts its internal clock from the time stamp to determine the data age. If the data age is less than the maximum age allowed, the data are applied; otherwise, the connection goes to the safety state. The device application is notified so that the connection safety state can be reflected accordingly.

The ping request-and-response sequence is repeated periodically to correct for any producer or consumer time base drift.

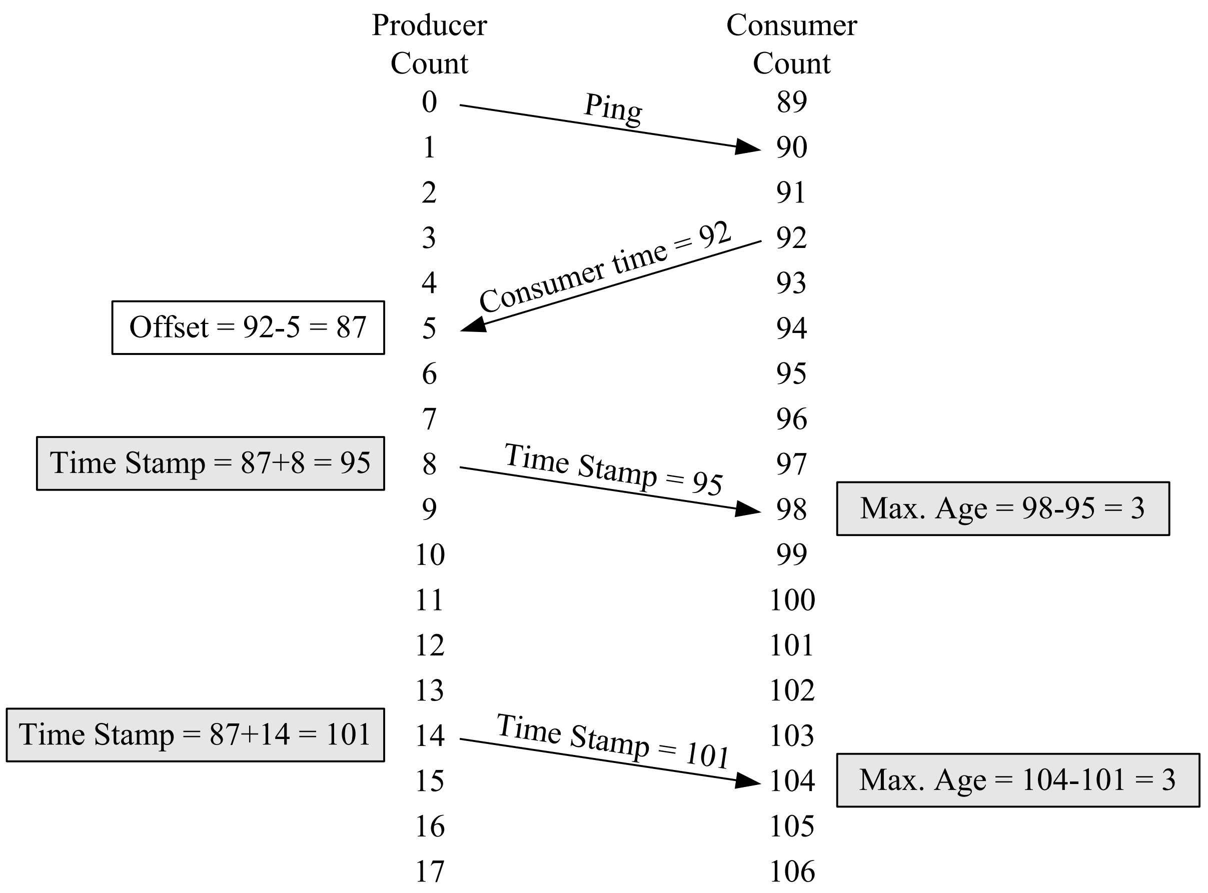


Figure 58 Time Stamp

5.2.3.2. Production Identifier (PID)

A Production Identifier is encoded in all data produced by a Safety Connection to ensure that each received message arrives at the correct consumer. The PID is derived from an electronic key, the device serial number and the CIP connection serial number. Any safety device inadvertently receiving a message with the incorrect PID will go to a safety state. Any safety device that does not receive a message within the expected time interval with the correct PID will also go to a safety state. This measure ensures that messages are routed correctly in multi-network applications.

5.2.3.3. Safety CRC (Cyclic Redundancy Code)

All safety transfers on CIP Safety use Safety CRCs to ensure the integrity of the transfer of information. The Safety CRCs serve as the primary means of detecting possible corruption of transmitted data. They provide detection up to a Hamming distance of four for each data transfer section, though the overall Hamming distance coverage is greater for the complete transfer due to the protocol's redundancy. The Safety CRCs are generated in the Safety Producers and checked in the Safety Consumers. Intermediate routing devices do not examine the Safety CRCs. Thus, by employing end-to-end Safety CRCs, the individual data link CRCs are not part of the safety function. This eliminates certification requirements for intermediate devices and helps to ensure that the safety protocol is independent of the network technology. The Safety CRC also provides a strong protection mechanism that allows detection of underlying data link errors, such as bit stuffing or fragmentation.

While the individual link CRCs are not relied on for safety, they are still enabled. This provides an additional level of protection and noise immunity by allowing data retransmission for transient errors at the local link.

5.2.3.4. Redundancy and Cross-check

Data and CRC redundancy with cross-checking provides an additional measure of protection by detecting possible corruption of transmitted data. By effectively increasing the Hamming distance of the protocol, these measures allow long safety data packets – up to 250 bytes – to be transmitted with high integrity. For short packets of two bytes or less, data redundancy is not required; however, redundant CRCs are cross-checked to ensure integrity.

5.2.3.5. Diverse Measures for Safety and Standard

The CIP Safety protocol is present only in safety devices, which prevents standard devices from masquerading as safety devices.

5.2.4. Safety Connections

CIP Safety provides two types of Safety Connections:

- Unicast Connections;
- Multicast Connections.

A Unicast Connection, as shown in Figure 59, allows a safety validator client to be connected to a safety validator Server using two link-layer connections

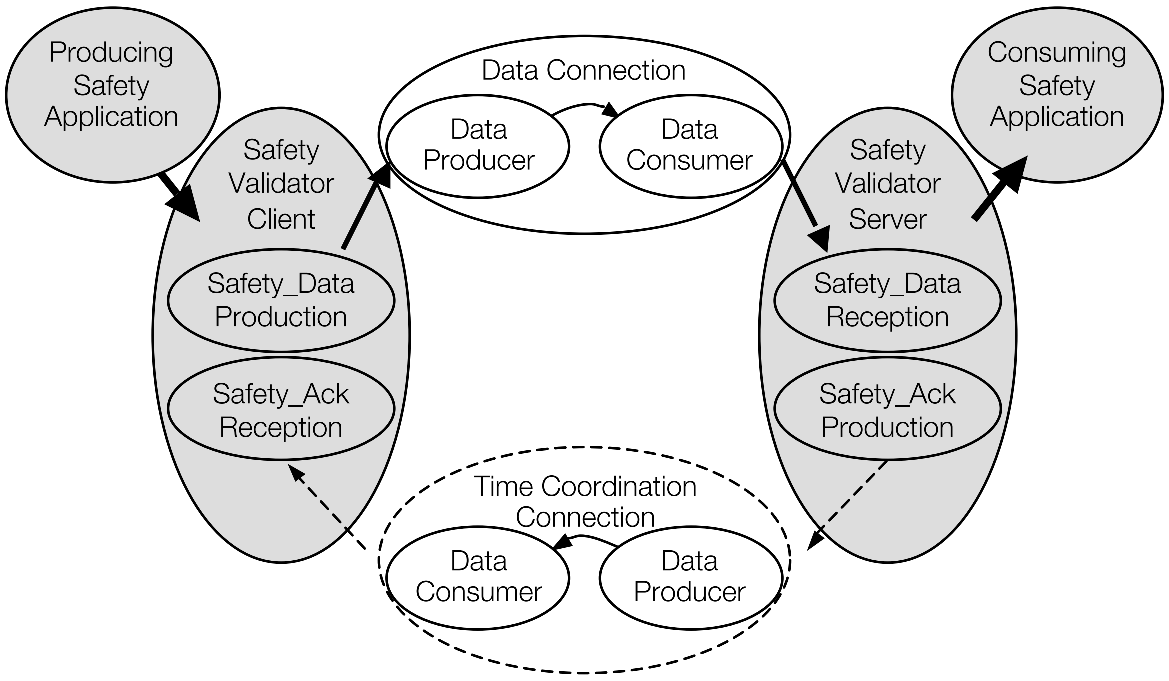


Figure 59 Unicast Connection

A Multicast Connection, as shown in Figure 60, allows up to 15 safety validator servers to consume safety data from a safety validator client. When the first safety validator server establishes a connection with a safety validator client, a pair of link layer connections are established, one for data-and-time correction and one for time coordination. Each new safety validator server uses the existing data-and-time correction connection and establishes a new time coordination connection with the safety validator client.

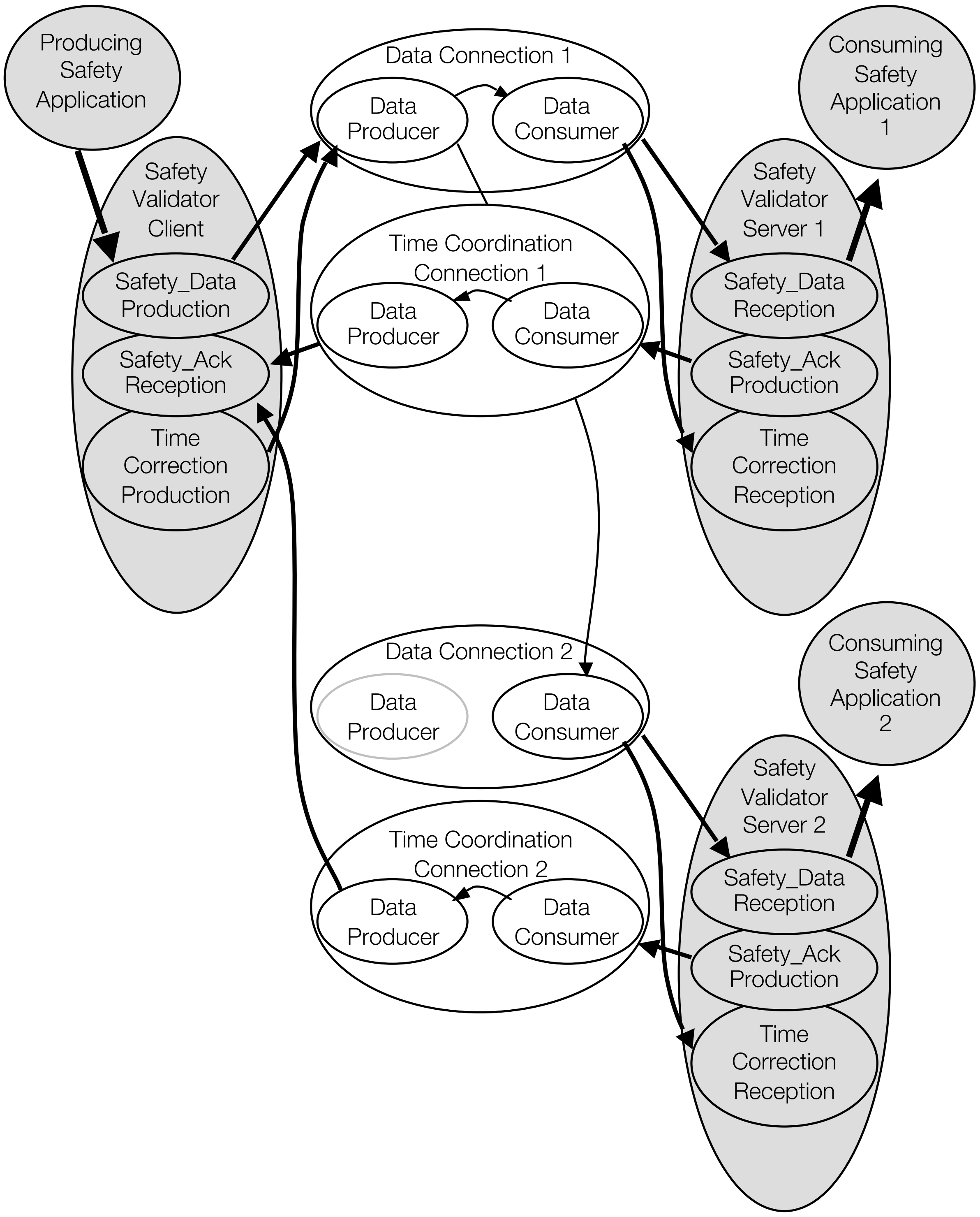


Figure 60 Multicast Connection

To optimize throughput on DeviceNet, each Multicast Connection uses three data link connections, as shown in Figure 61.

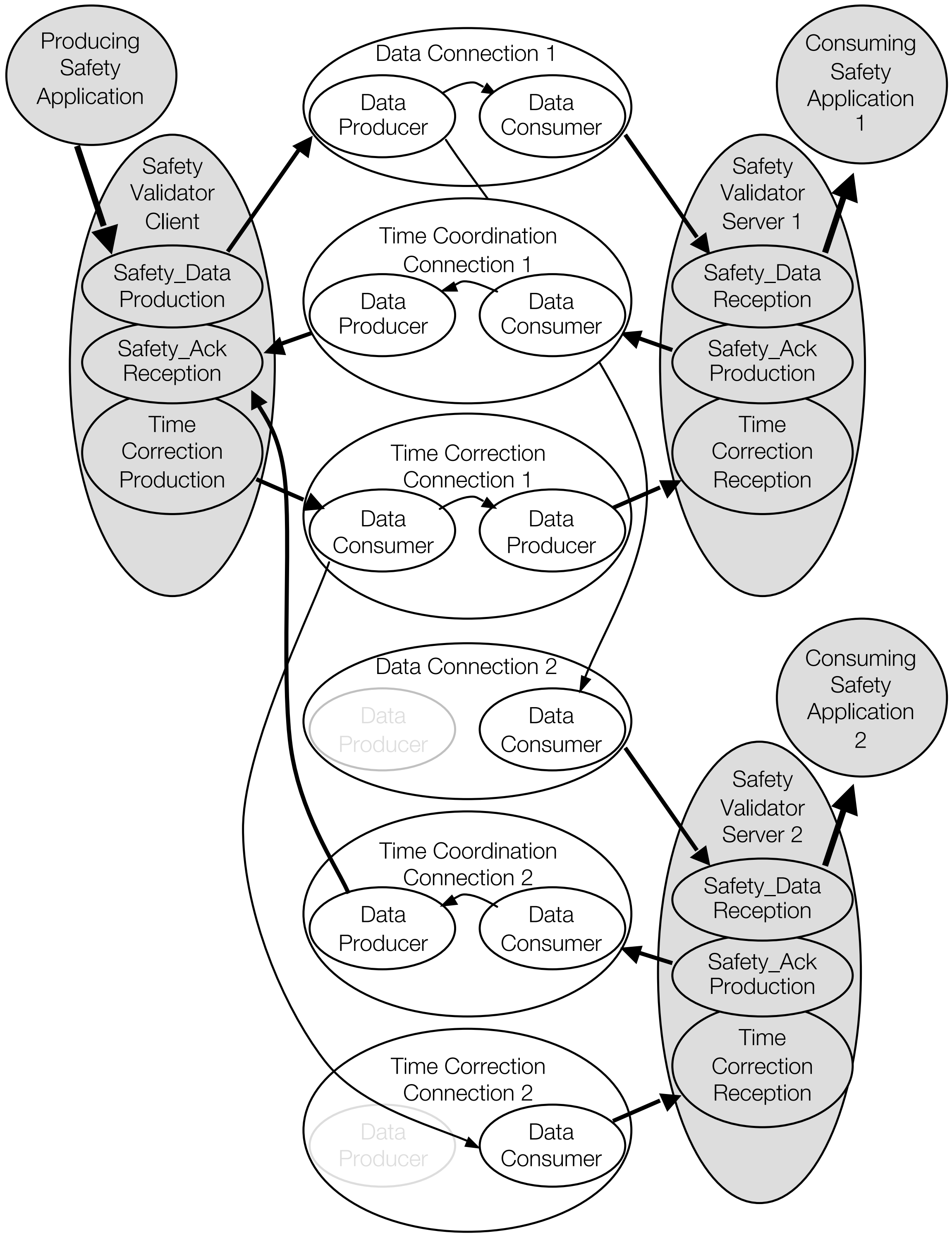


Figure 61 Multicast Connection on DeviceNet

CIP Safety implementations on other networks do not require this split. The data-and-time correction messages are sent on separate connections. This allows short messages to be transmitted on DeviceNet within a single CAN frame and reduces the overall bandwidth, since the time correction and time coordination messages are sent at much slower periodic intervals.

When multicast messages are routed off-link, the router combines the data-and-time correction messages from DeviceNet and separates them when messages reach DeviceNet. Since the safety message contents are unchanged, the router provides no safety function.

5.2.5. Message Packet Sections

CIP Safety has four message sections:

- Data;
- Time-stamp;
- Time correction;
- Time coordination.

The description of these formats goes beyond the scope of this book. For available materials on this topic that go into more detail see [5], [36].

5.2.6. Configuration

Before safety devices can be used in a safety system, they first must be configured and connections must be established. The process of configuration requires placement of configuration data from a configuration tool in a safety device. There are two possible sequences for configuration:

- Configuration tool directly to device;
- Via an intermediate device.

In the configuration tool-to-device case, as shown in Figure 62, the configuration tool writes directly to the device to be configured (1), (2).

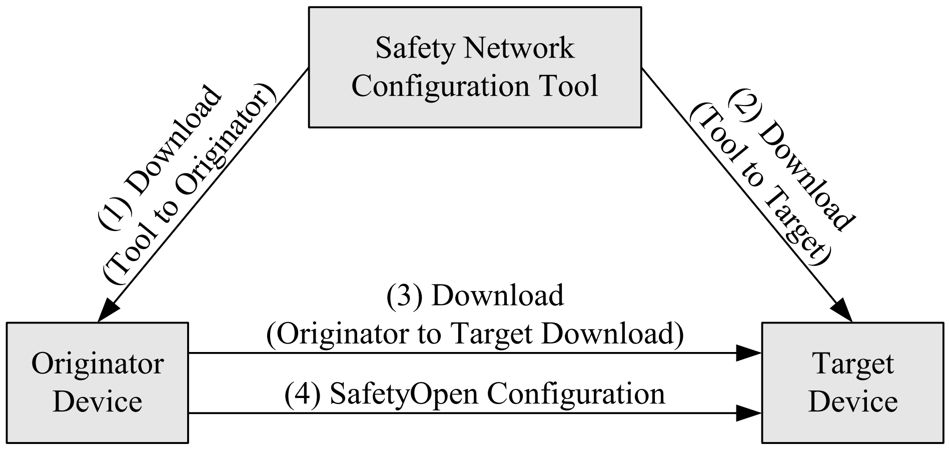

Safety Network Configuration Tool
(1) Download (Tool to Originator)
(2) Download (Tool to Target)
(3) Download (Originator to Target Download)
(4) SafetyOpen Configuration
Originator Device
Target Device

Figure 62 Configuration Transfers

In the case of intermediate device configuration, the tool first writes to an originator (1) and the originator writes to the target using an Originator-to-Target Download (3) or a SafetyOpen service (4). The SafetyOpen service (4) is unique in that it allows a safety connection to be established at the same time that a device is configured.

5.2.7. Connection Establishment

CIP provides a connection establishment mechanism, using a Forward_Open service that allows producer-to-consumer connections to be established locally or across multiple networks via intermediate routers. An extension of the Forward_Open, called the SafetyOpen service, has been created to allow the same multi-network connections for safety.

There are two types of SafetyOpen requests:

- Type 1: With configuration;
- Type 2: Without configuration.

With the Type 1 SafetyOpen request, configuration and connections are established at the same time, allowing rapid configuration of devices with simple and relatively small configuration data.

With the Type 2 SafetyOpen request, the safety device first must be configured and the SafetyOpen request then establishes a Safety Connection. This separation of configuration and connection establishment allows the configuration of devices with large and complex configuration data.

In both cases, the SafetyOpen request establishes all underlying link layer connections – across the local network as well as any intermediate networks and routers.

5.2.8. Configuration Implementation

CIP Safety provides the following protection measures to ensure configuration integrity:

- Safety Network Number;
- Password Protection;
- Configuration Ownership;
- Configuration Locking.

5.2.8.1. Safety Network Number

The Safety Network Number provides a unique network identifier for each network in the safety system. The Safety Network Number, combined with the local device address, allows any device in the safety system to be uniquely addressed.

5.2.8.2. Password Protection

All safety devices support the use of an optional password. The password mechanism provides an additional protection measure, prohibiting the reconfiguration of a device without the correct password.

5.2.8.3. Configuration Ownership

The owner of a CIP Safety device can be specified and enforced. Each safety device can specify that it be configured only by a selected originator, or that the configuration is accomplished by a configuration tool.

5.2.8.4. Configuration Locking

Configuration Locking provides the user with a mechanism to ensure that all devices have been verified and tested prior to being used in a safety application.

5.2.9. Safety Devices

The relationship of the objects within a safety device is shown in Figure 63. Note that CIP Safety extends the CIP object model, with the addition of Safety I/O assemblies, safety validator and Safety Supervisor Objects.

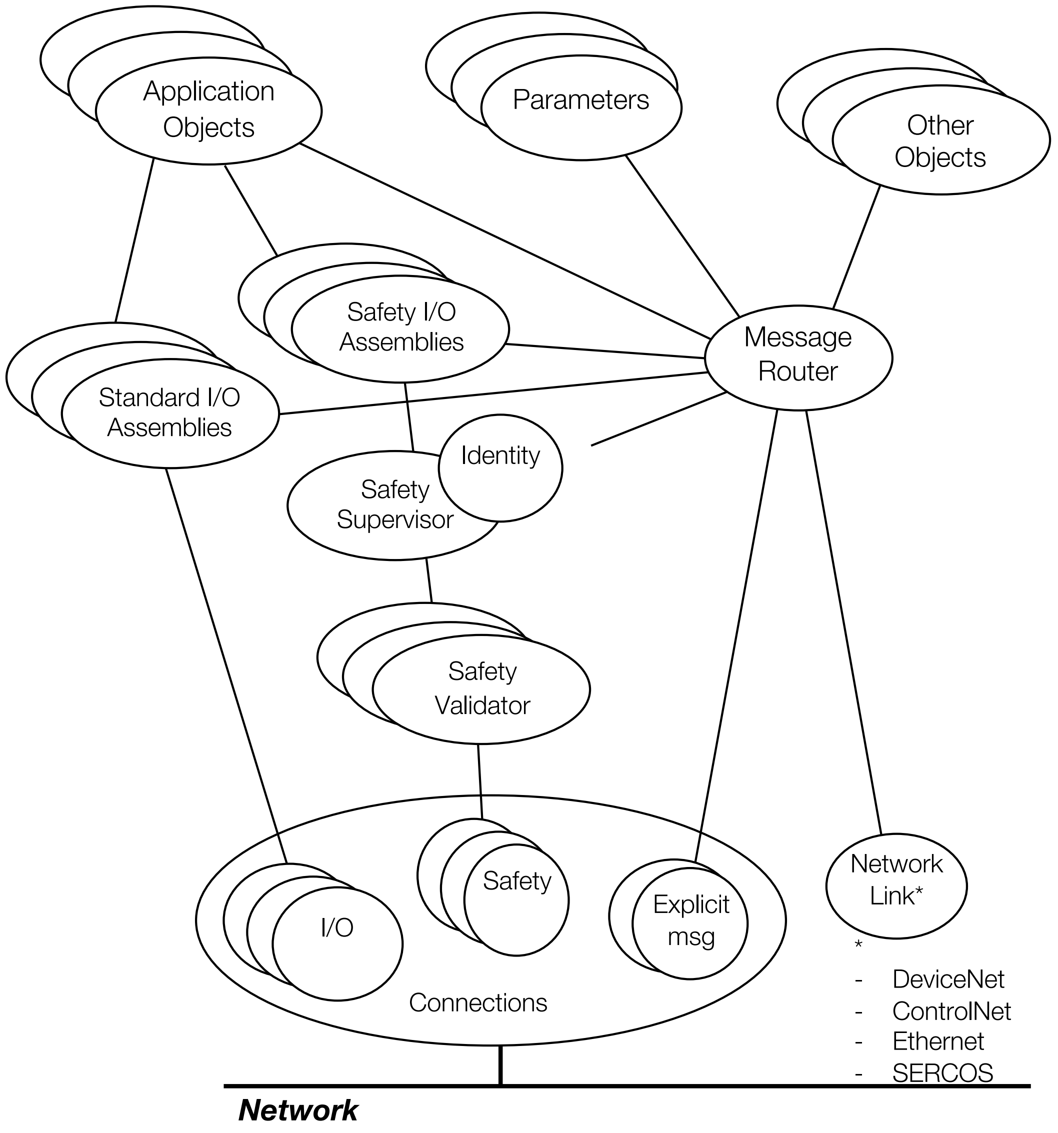


Figure 63 Safety Device Objects

5.2.10. Additional Objects

CIP Safety requires two additional objects, the Safety Supervisor Object and the Safety validator object.

5.2.10.1. Safety Supervisor Object, (Class ID: 0x39)

The Safety Supervisor Object provides a common configuration interface for safety devices. The Safety Supervisor Object centralizes and coordinates application object state behavior and related status information, exception status indications (alarms and warnings) and defines a behavior model which is assumed by objects belonging to safety devices.

5.2.10.2. Safety Validatior Object, (Class ID: 0x3A)

The Safety validator object contains the information necessary to coordinate and maintain reliable safety connections between client and server safety applications. The primary role of the Safety validator object is to act as a safety transport manager of multiple low-level CIP connections that together form a complete safety connection.

5.2.11. CIP Safety on Sercos

5.2.11.1. What is Sercos?

Sercos (**SE**rial **R**ealtime **CO**mmunication **S**ystem), the digital drive interface approved as international standard IEC 61491 [28] in 1995, is optimized for high-speed deterministic motion control, where the exact synchronization of multiple drives is required. Sercos has become a globally accepted real-time networking standard for demanding motion control applications over the last decade. Sercos has outstanding technical features like real time capabilities, high performance, noise immunity, and a very large variety of products and suppliers. Sercos not only defines the protocol structure, but also includes an ample variety of profile definitions (parameters and functionalities), which are already successfully used in a large number of applications. Sercos is supported and maintained by Sercos International [45].

The third generation Sercos (Sercos III) combines the proven mechanisms of Sercos interface with Ethernet's physics and protocol. Typical Sercos III networks use a double ring structure which provides media redundancy with fast switch-over. In addition to the ring structure, a linear structure is also possible, see Figure 64.

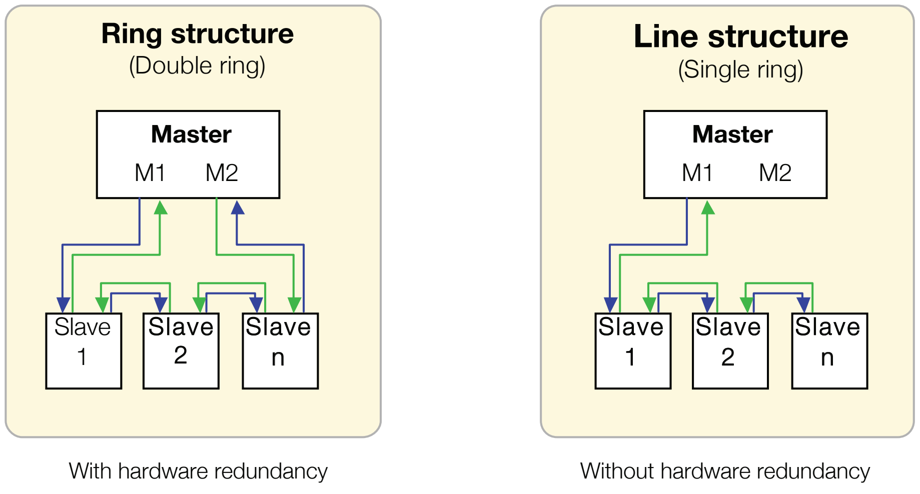


Figure 64 Sercos III Ring and Line Topology

A Sercos ring or line structure consists of one master and multiple slaves – drives, I/O, sensors. Multiple rings can be used in a network to realize distributed and hierarchical network structures. The communication is based on a time-slot protocol using fixed and distinct communication cycles. A communication cycle is divided into two channels with a timing control.

In the real-time channel, collective Sercos III telegrams are transferred as broadcast data. This increases the bandwidth and improves the protocol efficiency. The addressing of the Sercos III devices is achieved by pre-defined addresses or by addresses assigned by the master (remote addressing). Sercos III telegrams are processed on the fly to reduce delay times in a network.

In the non-real-time channel, any non-real-time Ethernet frames can be sent as individual telegrams to any device in the network. The addressing for this is carried out directly via the MAC addresses allocated to the master and slave devices.

5.2.11.2. CIP Safety on Sercos

CIP Safety on Sercos extends the functionality of the Sercos III real-time communication system to support both safety-relevant and non-safety relevant data transmission over one single network.

The CIP Safety Stack (CSS) is implemented corresponding to the CIP Safety specification without modifications. Thus, CIP Safety on Sercos uses the safe message format and the CIP Safety objects and services of the CIP Safety specification, as well as the same CRC polynomials and algorithms. For this reason, it is possible for the CIP Safety Adaptation Layer (CSAL) as well as the Sercos III subordinate communication system, including the Sercos Messaging Protocol (SMP), to be viewed as part of the non-safety-relevant transfer, see Figure 65.

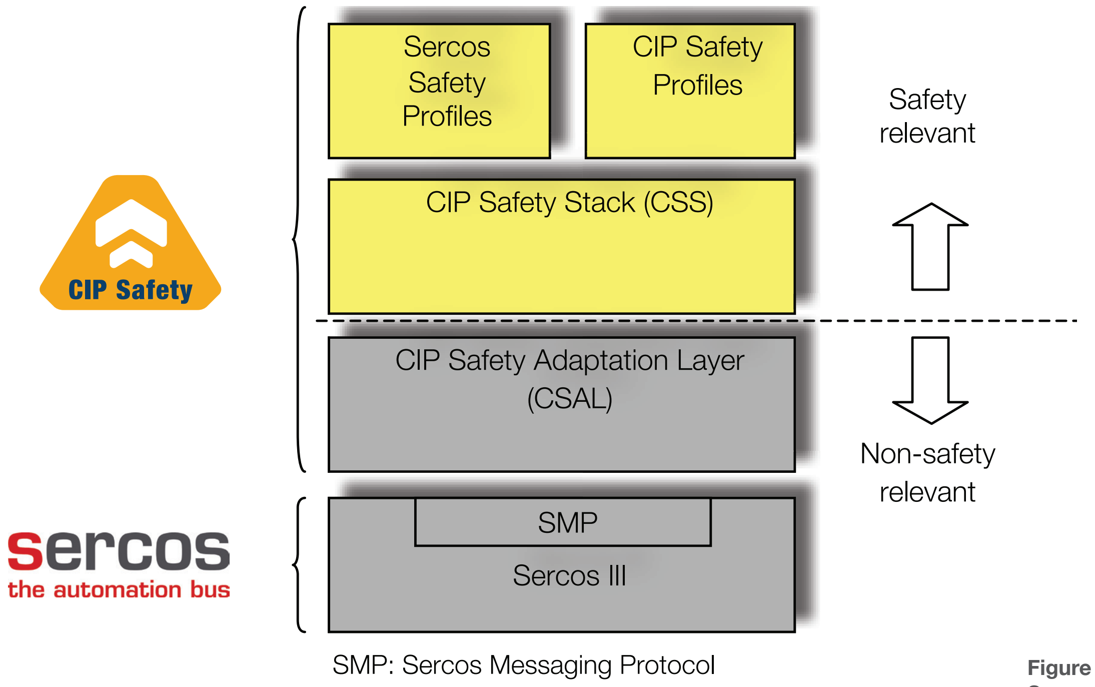


Figure 65 CIP Safety on Sercos - Structure

The main task of the CIP Safety Stack (CSS) in a CIP Safety Device is to create and process CIP Safety messages using cyclic process data, and to correspondingly specify communication errors with the help of the different error recognition measures. A CIP Safety Device can be a tool, an originator or a target. In addition to the process data, the CSS must also process configuration data.

Because Sercos III does not provide any transport connections, an adaptation layer is required in order to fill the gaps between CIP Safety and Sercos III. An important task is assigned to the Sercos Messaging Protocol (SMP), which is located above the cyclic Sercos III connections. It offers the CSAL services in order to send messages, which can be of any length, from a producer to the consumers. In order to transport this data, the SMP uses data containers with a fixed length in a cyclic connection. Messages that are larger than one data container are split up into several fragments and transferred subsequently. Messages with a higher priority can interrupt the transfer of long, fragmented messages with lower priority. This means that an effective multiplexing of several logical communication channels in a single transport container is possible. The SMP is part of the non-safety-relevant transfer. Mechanisms and measures for the data integrity of CIP Safety are not influenced.

With Sercos III and CIP Safety the implementation of a wide range of topologies is possible. The range stretches from structures with a central safety control to completely decentralized solutions without any safety control. With Sercos III and CIP Safety it is also possible to route safe data beyond the limits of a Sercos network, and also beyond non-safety-relevant participants.

CIP Safety on Sercos has been integrated into the CIP Specification by extending Volume 5 (CIP Safety) and through some minor adjustments to Volume 1 (CIP Common). Since the safety protocol was not changed at all, the main parts of Volume 5 were only changed in those areas where modified wording was needed to accommodate the additional network. However, there is a new appendix in Volume 5 (Appendix G) that describes how CIP Safety is being transported on the Sercos transport layer. Since the CIP Safety protocol as such remained unchanged, most of the adaptation work was done on the Sercos side which is reflected in the CIP Safety on Sercos specification available from Sercos International [45].

5.2.12. CIP Safety Summary

CIP Safety is a scalable, routable, network-independent safety protocol based on extensions to the CIP architecture. This concept can be used in solutions ranging from device-level networks such as DeviceNet to higher level networks such as EtherNet/IP. Designing network independence into CIP Safety allows multi-network routing of Safety Connections. Functions such as multi-network routing and multicast messaging provide a strong foundation that enables users to create the rapidly responding local cells and interconnect remote cells that are required for today's safety applications. CIP Safety's design also enables expansion to future network technologies as they become available.

5.3. CIP Energy

The optimization of energy usage is a natural expansion of ODVA's application coverage for industrial automation. The objects that support this functionality were added to Volume 1 of the CIP Networks Library in Edition 3.12. The management of energy usage methodology described in the specification defines a set of standard attributes, services and behaviors that will facilitate the reporting of industrial devices' use of operational energy, and the control of industrial devices into and out of non-operational energy conserving states.

Four objects have been defined as of the publication of this text. A "Base Energy" application object standardizes access to the most basic data and services common to the various energy resources used in industry. The base energy object also provides the means to aggregate energy information at various levels of the enterprise and present this data consistently at all levels: from the asset level at the bottom of the energy usage tree, up through the machine and process level to the system level and ultimately to the production and enterprise domains. Two resource type application objects are defined. An electrical energy object provides electrical energy data reporting capabilities and diagnostics for the electrical energy consumers and producers found within the various levels of an industrial facility. A “non-electrical energy object” provides unified reporting of energy consumption and production of non-electrical energy data such as gas and steam. A “power management object” provides standardized attributes and services to support the control of devices into and out of paused or sleep states. An overview of these objects in the overall construct of CIP is shown in Figure 66.

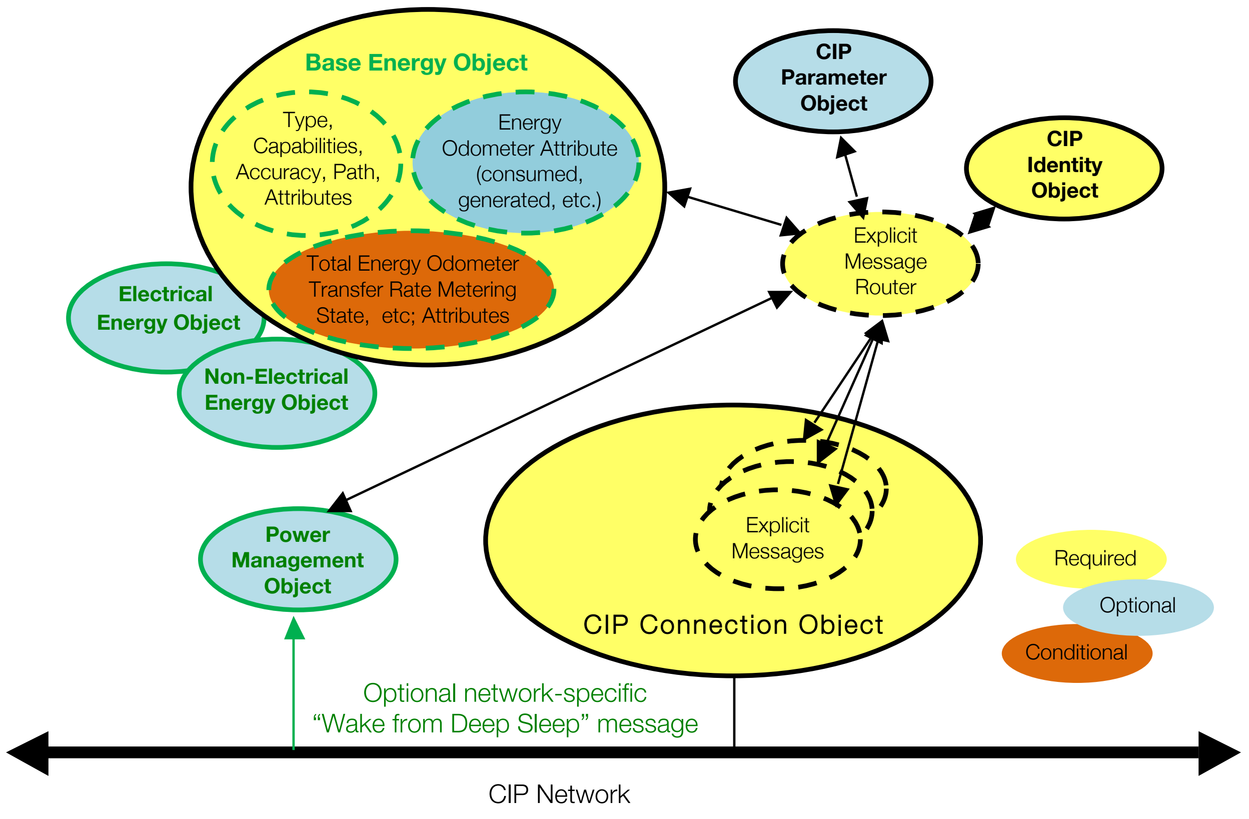

Base Energy Object
Type, Capabilities, Accuracy, Path, Attributes
Energy Odometer Attribute (consumed, generated, etc.)
Total Energy Odometer Transfer Rate Metering State, etc; Attributes
Electrical Energy Object
Non-Electrical Energy Object
CIP Parameter Object
CIP Identity Object
Explicit Message Router
Explicit Messages
CIP Connection Object
Power Management Object
Required
Optional
Conditional
Optional network-specific “Wake from Deep Sleep” message
CIP Network

Figure 66 CIP Energy Objects

What follows is a brief introduction to the objects involved in CIP Energy. More information can be found in references [63], [64].

5.3.1. Additional Objects

4 additional objects have been defined for energy management so far, the base energy object, the electrical energy object, the non-electrical energy object and the power management object.

5.3.1.1. Base Energy Object (Class ID: 0x4E)

The base energy object acts as an “Energy Supervisor” for energy implementations in CIP. It provides energy mode services, and can provide aggregation services for aggregating energy values up through the various levels of an industrial facility. It also provides a standard format for reporting energy metering results. The object is independent of the energy type and allows data and functionality specific to the energy type to be integrated into an energy system in a standard way.

Multiple instances of the base energy object may exist in a device. For instance, an electric power monitor may count metering pulse output transitions of a separate metering device. The count of such transitions, represented by a base energy object instance, would reflect the energy consumption measured by the separate device. As another example, a device may act as a proxy for the energy consumed by a number of simple devices, where each device is associated with a separate instance of the base energy object. An instance of the base energy object may exist as a stand-alone instance, or it may exist in conjunction with an instance of either the electrical energy object or the Non-electrical energy object. If an instance of either the electrical or non-electrical energy object is implemented in a device, it must be associated with a base energy object instance in the device (i.e., it is a child of the base energy object instance).

The object definition allows for creating five types of devices, Energy Measured, Energy Derived, Energy Proxy, Energy Rate Fixed and Energy Aggregated devices.

- Energy Measured devices are devices such as a power monitor that measures voltage, current, phase angle, etc., and calculates power and energy;

- Energy Derived devices are devices such as an overload relay that measures motor current, assumes a value of motor voltage and then derives the value of power and energy;

- Energy Proxy devices such as a controller with discrete outputs that control external devices, each of which has an associated user-provided energy transfer rate that the controller uses to calculate the energy used based on the state of the output;

- Energy Rate Fixed devices are simple devices that report a nominal or user-defined energy value when operating and zero when in a non-operating state:

- Energy Aggregated devices are devices that can collect energy usage of “child” devices and report them together as an aggregate value.

Figure 67 illustrates these devices.

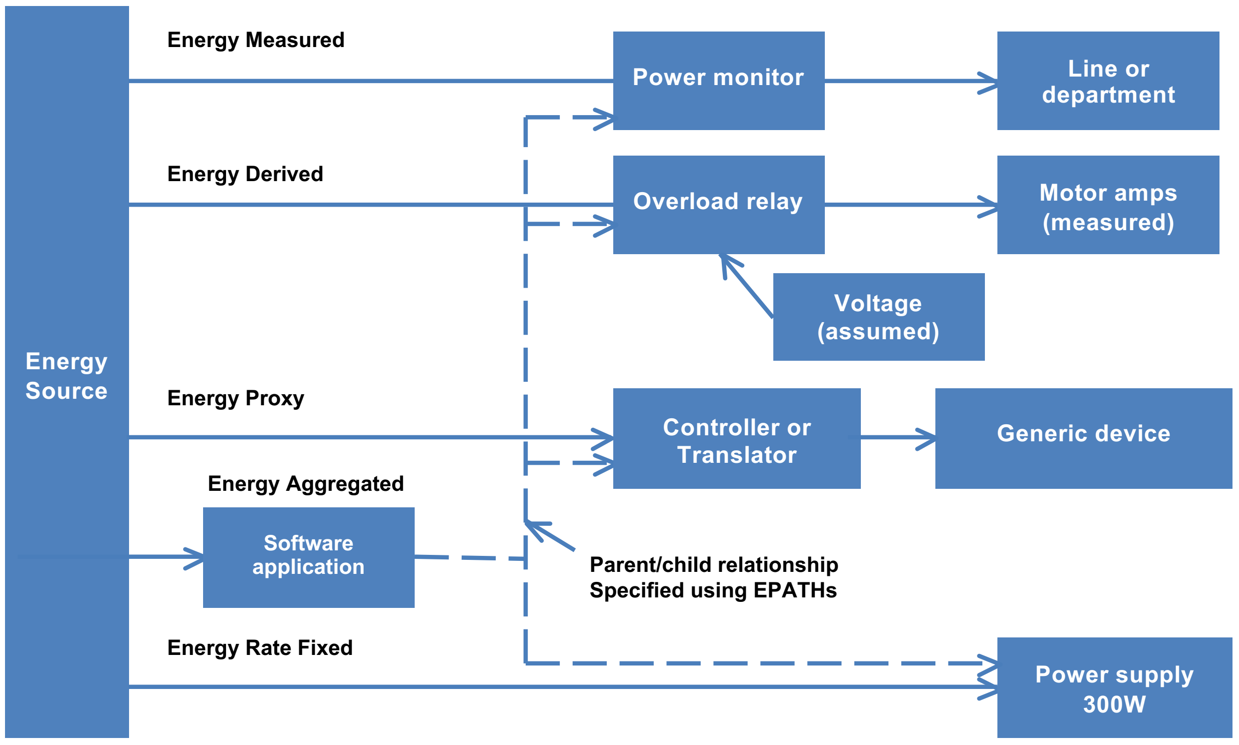


Figure 67 Types of Energy-Related Devices

Energy Odometers are defined to report large amounts of energy usage in a manner similar to a car odometer or a typical power meter on a residential home. The values are in multiples of kilowatt-hours as shown in Figure 68:

Array[4]	Array[3]	Array[2]	Array[1]	Array[0]
Terawatt-hours (kWh x 109)	Gigawatt-hours (kWh x 106)	Megawatt-hours (kWh x 103)	Kilowatt-hours (kWh)	Watt-hours (kWh x 10-3)

Figure 68 Energy Odometer

The Base Energy object may contain information about Electrical and Non-Electrical Energy object instances that it refers to or it may be a stand-alone instance with no children.

5.3.1.2. Electrical Energy Object (Class ID: 0x4F)

The electrical energy object provides unified electrical energy specific data reporting and diagnostics for the CIP enabled devices and processes found within the various levels of an industrial facility. Energy management is typically related to the measurement and reporting of a variety of metering results. This object provides for the consistent reporting of electrical energy data. Electrical energy is organized in a separate object to accommodate its alternating and poly-phase characteristics, which result in a collection of attributes that are unique among energy sources.
Using the electrical energy object in association with an instance of the base energy object (via the associated base energy object path, attribute 41) provides a comprehensive approach to reporting usage of electrical energy in a consistent and open fashion.

5.3.2. Non-Electrical Energy Object (Class ID: 0x50)

The non-electrical energy object provides unified non-electrical energy specific data reporting and diagnostics for the CIP enabled devices and processes found within the various levels of an industrial facility. Energy management is typically related to the measurement and reporting of a variety of metering results. This object provides for the consistent reporting of non-electrical energy data, including, without limitation, natural gas, fuel oil, steam, compressed air, hot water, chilled water, etc.

Using the non-electrical energy object in association with an instance of the base energy object (via the Associated base energy object Path, attribute 41) provides a comprehensive approach to reporting usage of non-electrical energy in a consistent and open fashion.

5.3.4. Power Management Object (Class ID: 0x53)

The power management object provides standardized attributes and services to support the control of devices into and out of paused or sleeping states. A device supporting the power management object can transition between various energy-related states. A Power Management service and optional adaptation specific sleep mechanisms are used to control entry into and exit from energy saving states. Within the paused states, a device may have multiple internal energy saving modes, each with a different power consumption level.
There are six basic energy-related states for CIP Power Management capable devices:

1. Power Off;
2. Not Owned – Device is operational but no client “owns” the Power_Management service of the device;
3. Owned – The device is operational and a client “owns” the Power_Management service of the device;
4. Paused – Energy saving state; CIP communication continues;
5. Sleeping – Energy saving state; CIP communication is suspended.;
6. Resuming – Device is transitioning from Paused to Owned.

Within a particular power management state, a device’s operational capabilities and power levels may remain in transition for a time until the agreed power usage level is attained.

5.4. Integration of non-CIP Networks

5.4.1. Integration of Modbus into CIP

5.4.1.1. Overview

With the advent of the Modbus translator customers can take advantage of EtherNet/IP and Modbus capabilities in the same network. The simplicity of the Modbus protocol is combined with the unique values of EtherNet/IP. The relationship is further strengthened because the Modbus/TCP protocol and EtherNet/IP are both based on standard Ethernet technology. Both protocols can coexist in the same network because they both operate over standard Ethernet. Migration of existing products is enabled with no custom hardware required.

By establishing the Modbus Integration SIG within ODVA, the seamless connection between the Modbus protocol and EtherNet/IP was enabled. Volume 7, Integration of Modbus Devices into CIP [7], was developed within the SIG and later approved by the ODVA. Edition 1.0 of Volume 7 supports translation between EtherNet/IP and Modbus/TCP. The Modbus Serial protocol support was added to Volume 7 in Edition 1.1 and approved by the ODVA at the beginning of 2008. Clarifications and updates to Volume 7 are being considered to continually improve the volume.

The Modbus protocol and the EtherNet/IP protocol make up the majority of the installed Ethernet-based device level products to date. Both are widely accepted standards with strong international organizations behind them and solid membership and participation from those organizations. Volume 7 links the two protocols together.

The Modbus Integration SIG was formed in May of 2007 with the purpose to create the Modbus translator. More than 20 different companies belong to the Modbus Integration SIG. The diverse membership of the SIG insured an unbiased specification was developed. The diversity of the SIG membership brought all perspectives into the development of the specification from the Modbus target device developer to the Programmable Logic Controller (PLC) manufacturer to the EtherNet/IP target vendor to the CIP originator designer. All aspects of the CIP and Modbus networks were and are represented.

The development of Volume 7 came with the requirement that the use of the Modbus translator would not force changes to existing Modbus target devices or the EtherNet/IP target devices. Also a requirement for the development was to minimize the impact to CIP originators. These requirements were met. The impact to the CIP originator was focused on the need to support both the Modbus/TCP and EtherNet/IP protocol on the same physical Ethernet port should a customer desire to do so.

The Modbus translator can be implemented in the CIP originator or as a CIP router, i.e. the translator can be in a PLC or as a standalone device. The Modbus translator effort and consequently Volume 7 of the CIP suite of protocols was targeted at CIP originator developers, CIP router developers and Modbus vendors. The first two audiences are obvious. The Modbus device vendor is targeted so that this vendor can better understand how their device can be more easily integrated into the CIP to Modbus solution.

As the name states the Modbus translator translates CIP objects and services into Modbus messages and function codes. A CIP originator communicates with and controls Modbus target devices through the Modbus translator. To the CIP originator the Modbus target devices appear as CIP target devices. The Modbus target devices believe they are being controlled by a Modbus client. The translation is transparent.

The user would place a standalone Modbus translator in the user's CIP-based network then connect Modbus target devices to the Modbus side of the translator. The other side of the translator would be connected to the CIP originator.

The user also has the option to use a Modbus translator as a module inside the CIP originator. In this case the user would talk EtherNet/IP across the backplane to the translator. The Modbus target devices connect externally to the PLC-based translator.

A third approach places the Modbus translator and the Modbus/TCP target devices on the same network as the EtherNet/IP target devices using the shared cabling. The EtherNet/IP traffic destined for the Modbus target devices would be sent to the Modbus translator and the Modbus translator would send the Modbus/TCP traffic over the same network to the Modbus/TCP target devices. The Modbus/TCP target devices would reply to the Modbus translator then the translator will translate the Modbus traffic into EtherNet/IP traffic sent to the CIP originator.

In all three cases, the CIP-to-Modbus translation allows communication using I/O connections as well as explicit messaging. When explicit messaging is used, the translator may offer two types of access to the data in the Modbus device, through publicly defined instances for Modbus data in the assembly object using Common Services or through the Modbus Object using object-specific services.

Logically in the OSI model of the TCP/IP stack, the Modbus translation on Ethernet sits above Modbus/TCP application in the application layer (Layer 7) and below the CIP application layers. The Modbus translator on Ethernet is on the same level as the EtherNet/IP encapsulation. The lower layers of the stack are the standard TCP/IP stack layering.

Changes to the CIP originator were limited. Port types were added for Modbus/TCP and Modbus Serial devices with associated port names and numbering. The identity object was extended to include Modbus-specific information and a Modbus Object was developed and included in the CIP Object Library. Status codes were added to identify errors and status items that are specific to Modbus. Other status codes were modified to better support Modbus and EtherNet/IP together.

All the capabilities of Modbus translator are detailed in Volume 7, "Integration of Modbus Devices into CIP" within the CIP suite of protocols. The updated Volume 1, Common Industrial Protocol (CIP) Specification, includes support for the Modbus translation as does the updated Volume 2, EtherNet/IP Adaptation of CIP.

Customers now have greater opportunity to improve their automation networks by being able to incorporate Modbus devices into their existing CIP networks seamlessly, especially in the case of EtherNet/IP and Modbus TCP devices on the same physical network. Customers have a broader range of products available to them and can mix and match features they need from a device without being restricted to a single protocol. Risk to the buyer is reduced by the availability of greater choices to use devices based on EtherNet/IP and Modbus/TCP.

5.4.1.2. Object Extensions

The definition of the Modbus translation has resulted in extensions of existing objects as well as the creation of new objects.

5.4.1.2.1. Extension of Existing Objects

The main extension for the Modbus integration is within the identity object which has been enhanced with an attribute containing Modbus-specific identity information plus a translation definition for standard CIP ID information.

A set of assembly object and parameter object Instances has been defined that mirrors Modbus-specific data tables (holding registers, input registers, coils, discrete inputs). These instance numbers are now represented by 32-bit numbers (UDINT) to accommodate the 4 data set ranges represented by 16-bit numbers each.

5.4.1.3. Additional Objects

Volume 7 defines two additional objects that are found only on Modbus devices, the Modbus object and the Modbus serial link object.

5.4.1.3.1. Modbus Object (Class ID: 0x44)

The Modbus object provides an interface to Modbus data existing in a CIP device. Within a CIP to Modbus translator the Modbus object provides an interface to the data and functions within a target Modbus device.

No instance attributes are defined for this object; instead, the Modbus object defines services that mirror Modbus Function Codes along with their associated address data.

5.4.1.3.2. Modbus Serial Link Object (Class ID: 0x46)

The Modbus serial link object is used for configuration of a Modbus serial data communication channel and includes link-specific counters and status information for the port. Each instance of this object represents the client portion of a Modbus serial channel, which allows the Modbus translator to read/write data with external Modbus serial servers.

5.4.2. Integration of IO-Link into CIP

IO-Link, standardized as IEC 61131-9, is a communication standard that extends the 24 VDC I/O interface of IEC 61131-2 to allow serial communication between I/O devices (IO-Link Masters) and sensors and actuators (IO-Link Devices).

An ODVA SIG, established in fall 2012, is working on the definition of the integration of IO-Link masters and devices into CIP as of the publication of this document. Check www.odva.org for updates on this work.

6. Conformance Testing

Open specifications, such as those managed by ODVA, both provide vendors with the ability to build products that will interoperate with products from other vendors, and allow users to choose products that will interoperate in multi-vendor systems by ensuring a common network interface for given device types. In order to achieve interoperability of devices from multiple vendors, product compliance with these open specifications is essential.

ODVA drives product compliance with the CIP Network Specifications primarily in two ways. First, each vendor is required to sign a Terms of Usage Agreement for the ODVA technology or technologies, for which they intend to make, have made, sell or have sold products. In signing this agreement, the vendor agrees to comply with the network technology specification and meet a set of user responsibilities, including the conformance testing of developed and/or sold products. A list of authorized vendors can be found on the ODVA web site.

Second, ODVA administers a vendor-independent conformance testing process. The goal of the ODVA conformance testing process is to help to ensure, to the greatest extent practicable, that products implementing ODVA technologies and standards comply with the ODVA specifications and interoperate in multi-vendor systems.

A cornerstone of this process is the successful completion of the ODVA conformance test at an ODVA authorized test service provider (TSP). A full list of ODVA authorized TSPs can be found on the ODVA website. TSPs perform conformance tests that are designed, developed and managed by ODVA and conduct the tests in accordance with ODVA test requirements and procedures. ODVA TSPs must meet certain standards, including vendor-independence, neutrality and technical competency in networks and testing practices. The ODVA conformance test is typically a composite test comprised of three parts:

- An automated software test that verifies the function of the network protocol. Depending on the complexity of the device, several thousand messages are transmitted to the device under test (DUT). To ensure a test that is closely adapted to the characteristics of the DUT, the manufacturer must provide a formal description of all relevant CIP features of the DUT;

- A hardware test that examines the characteristics of the physical layer for conformance. Physical layer tests vary by network and may include product labeling, indicator operation, isolation, connectors, mis-wiring, voltage ranges, timing, etc.;

- An interoperability test that exercises the product using prescribed test scenarios designed to demonstrate the successful interoperability of the product in multi-vendor systems.

The automated conformance test software is a Windows®-based tool that uses a network interface card in the PC to access the device under test. It is recommended that device developers run this test in their own lab before taking devices to a test service provider. The hardware test (where appropriate) and the system interoperability test involve more complex test setups that typically are not available to device developers but are documented in the test plans or other ODVA publications. The vendor of the product may, at its option, observe the test at the TSP.

Upon the product's successful completion of the test, the TSP submits the test results to ODVA for review and final approval. Contingent on passing results from the conformance tests and other requirements of ODVA, ODVA issues a Declaration of Conformity for the product. Declarations of Conformity are posted on ODVA's website at www.odva.org.

Adjunct tests are also available from the ODVA headquarters' test service provider. These adjunct tests require that the device submitted for additional testing passes the appropriate conformance test first. Adjunct tests include:

- DeviceNet Semiconductor Industry (test is in addition to DeviceNet Node test);

Passing adjunct test results are listed on the Declaration of Conformity for the device or device family.

Products that have received an official Declaration of Conformity from ODVA earn the right to use ODVA's CONFORMANT certification marks as appropriate for the network connectivity of the product. (Refer to the ODVA Identity Guidelines on the ODVA web site for more information on logo usage.) End users should check the ODVA web site under "Product Compliance" for the list of ODVA issued Declarations of Conformity or look the following conformance mark on a product:

ODVA™
CONFORMANT

7. Endnotes

[1] CIP Networks Library, Volume 1, Common Industrial Protocol, Edition 3.18, April 2015, © 2001 through 2015 ODVA, Inc.

[2] CIP Networks Library, Volume 2, EtherNet/IP Adaptation of CIP, Edition 1.19, April 2015, © 1999 through 2015 ODVA, Inc.

[3] CIP Networks Library, Volume 3, DeviceNet Adaptation of CIP, Edition 1.14, November 2013, © 1994 through 2013 ODVA, Inc.

[4] CIP Networks Library, Volume 4, ControlNet Adaptation of CIP, Edition 1.8, April 2013, © 1997 through 2013 ODVA, Inc.

[5] CIP Networks Library, Volume 5, CIP Safety, Edition 2.11, April 2015, © 2005 through 2015 ODVA, Inc.

[6] CIP Networks Library, Volume 6, CompoNet Adaptation of CIP, Edition 1.7, April 2010, © 2006 through 2010 ODVA, Inc.

[7] CIP Networks Library, Volume 7, Integration of Modbus Devices into the CIP Architecture, Edition 1.7, April 2014, © 2007 through 2014 ODVA, Inc.

[8] Planning and Installation Manual, DeviceNet Cable System, Publication number PUB00027, downloadable from ODVA website (http://www.odva.org/).

[9] Recommended IP Addressing Methods for EtherNet/IP Devices, Publication number PUB00028, downloadable from ODVA website (http://www.odva.org/).

[10] Recommended Functionality for EtherNet/IP Devices, Publication number PUB00070, downloadable from ODVA website (http://www.odva.org/).

[11] EtherNet/IP Interoperability Test Procedures, Publication number PUB00095, downloadable from ODVA website (http://www.odva.org/).

[12] Performance Test Terminology for EtherNet/IP Devices, Publication number PUB00080, downloadable from ODVA website (http://www.odva.org/).

[13] Performance Test Methodology for EtherNet/IP Devices, Publication number PUB00081, downloadable from ODVA website (http://www.odva.org/).

[14] EtherNet/IP Quick Start for Vendors Handbook – A Guide for EtherNet/IP Developers, Publication number PUB00213, downloadable from ODVA website (http://www.odva.org/).

[15] EtherNet/IP Media Planning and Installation Manual, Publication number PUB00148, downloadable from ODVA website (http://www.odva.org/).

[16] IEC 61131-2:2007 – Programmable controllers – Part 2: Equipment requirements and tests.

[17] IEC 61131-3:2013 – Programmable controllers – Part 3: Programming languages.

[18] Controller Area Network – Basics, Protocols, Chips and Application. IXXAT Automation, 2001. ISBN 3-00-007376-0.

[19] ISO 11898:1993 – Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication.

[20] CAN Specifications – Controller Area Network, Version 2.0, 1991, Robert Bosch GmbH. http://www.semiconductors.bosch.de/pdf/can2spec.pdf.

[21] IEEE 802.3:2000, ISO/IEC 8802-3:2000 – IEEE Standard for Information technology – Local and metropolitan area networks – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specification.

[22] IEEE 802.3:2002 – Information Technology – Telecommunication & Information Exchange Between Systems – LAN/MAN – Specific Requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications 2002.

[23] ISO/IEC 7498-1:1994 – ISO/IEC Standard – Information technology – Open Systems Interconnection – Basic Reference Model.

[24] IEC 61508:1998 – Functional safety of electrical/electronic/programmable electronic safety-related systems.

[25] IEC 62026-3:2014 – Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 3: DeviceNet.

[26] IEC 62026-7:2010 – Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 7: CompoNet.

[27] IEC 61784-5-2:2013 – Industrial communication networks – Profiles – Part 5-2: Installation of fieldbuses – Installation profiles for CPF 2.

[28] IEC 61491:1995 – Electrical equipment of industrial machines – Serial data link for real-time communications between controls and drives.

[29] EN 62026-3:2015 – Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 3: DeviceNet.

[30] Chinese national standard: GB/T 18858:2003 – Low-voltage Switchgear and Controlgear Controller-Device Interface.

[31] IEC 61158:2014 – Digital data communications for measurement and control – Network for use in industrial control systems.

[32] Draft proposal test and certification guideline, safety bus systems, BG Fachausschuß Elektrotechnik 28-May-2000.

[33] EN 50159-1:2001 – Railway applications, communication, signaling and processing systems.

[34] GS-ET-26 – Grundsatz für die Prüfung und Zertifizierung von „Bussystemen für die Übertragung sicherheitsrelevanter Nachrichten“, Fachausschuss "Elektrotechnik" Prüf- und Zertifizierungsstelle im BG-PRÜFZERT, downloadable from http://www.bgetf.de/bilder/pdf/gs-et-26_a05-2002.pdf.

[35] ISO 13849-1:2006 – Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design.

[36] David A. Vasko, Suresh R. Nair: CIP Safety: Safety Networking for the Future; Proceedings of the 9th International CAN Conference, Munich, Germany. 2003. Downloadable from http://www.can-cia.org/

[37] Hideki Harada, "CompoNet: Innovations for High Performance Sensor-Actuator Applications", CIP Networks Conference & 12th Annual Meeting, Englewood, California. Presented Papers, 2007, downloadable from ODVA website (http://www.odva.org/).

[38] Hedon Ri, Overview of CompoNet Developer’s Toolkit, Presented at the ODVA 2007 CIP Networks Conference & 12th Annual Meeting, 2007, Englewood, California. Downloadable from ODVA website (http://www.odva.org/).

[39] Hedon Ri, Design Considerations for CompoNet Processors, Presented at the ODVA 2007 CIP Networks Conference & 12th Annual Meeting, 2007, Englewood, California. Downloadable from ODVA website (http://www.odva.org/).

[40] IEEE 1588-2008 – Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.

[41] Viktor Schiffer: Modular EDSs and other EDS Enhancements for DeviceNet; Proceedings of the 9th International CAN Conference 2003.

[42] Viktor Schiffer: Device Configuration Using Electronic Data Sheets, ODVA 2003 Conference and 9th Annual Meeting, Ann Arbor, Michigan.

[43] Viktor Schiffer, Ray Romito: DeviceNet Development Considerations, 2000.

[44] ControlNet Product Developer’s Guide, Rockwell Automation, Publication Number 9220-6.5.1, downloadable from http://literature.rockwellautomation.com.

[45] Sercos International, http://www.sercos.de/EUROPE-English.15.0.html.

[46] FDT Group Homepage. http://www.fdtgroup.org/.
[47] RFC 768 – User Datagram Protocol, 1980.
[48] RFC 791 – Internet Protocol, 1981.
[49] RFC 792 – Internet Control Message Protocol, 1981.
[50] RFC 793 – Transmission Control Protocol, 1981.
[51] RFC 826 – Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, 1982.
[52] RFC 894 – Standard for the transmission of IP datagrams over Ethernet networks, 1984.
[53] RFC 1103 – Proposed standard for the transmission of IP datagrams over FDDI Networks, 1989.
[54] RFC 1112 – Host extensions for IP multicasting, 1989.
[55] RFC 1122 – Requirements for Internet Hosts – Communication Layers, 1989.
[56] RFC 1123 – Requirements for Internet Hosts – Application and Support, 1989.
[57] RFC 1127 – Perspective on the Host Requirements RFCs, 1989.
[58] RFC 1171 – Point-to-Point Protocol for the transmission of multi-protocol datagrams over Point-to-Point links, 1990.
[59] RFC 1201 – Transmitting IP traffic over ARCNET networks, 1991.
[60] RFC 1392 – Internet Users' Glossary.
[61] RFC 2236 – Internet Group Management Protocol, Version 2, 1997.
[62] RFC 5227 – IPv4 Address Conflict Detection.

All RFCs are downloadable from http://www.faqs.org/rfcs/.

[63] ODVA Technology at a Glance, Technical Approach to Optimization of Energy Usage, Publication number PUB00265, available through ODVA website (http://www.odva.org/).
[64] ODVA white paper, Optimization of Energy Usage (OEU™): ODVA's Vision of Energy Optimization for the Industrial Consumer, Publication number PUB00246, downloadable from ODVA website (http://www.odva.org/).

8. Bibliography

CIP Networks Library, Volume 1, Common Industrial Protocol, Edition 3.18, April 2015, © 2001 through 2015

CIP Networks Library, Volume 2, EtherNet/IP Adaptation of CIP, Edition 1.19, April 2015, © 1999 through 2015

CIP Networks Library, Volume 3, DeviceNet Adaptation of CIP, Edition 1.14, November 2013, © 1994 through 2013

CIP Networks Library, Volume 4, ControlNet Adaptation of CIP, Edition 1.8, April 2013 © 1997 through 2013

CIP Networks Library, Volume 5, CIP Safety, Edition 2.11, April 2015, © 2015 ODVA.

9. Related Publications

Planning and Installation Manual, DeviceNet Cable System, Publication number PUB00027, downloadable from ODVA website (http://www.odva.org/)

EtherNet/IP Planning and Installation Manual, Publication number PUB00148, downloadable from ODVA website (http://www.odva.org/)

Performance Test Terminology for EtherNet/IP Devices, Publication number PUB00080, downloadable from ODVA website (http://www.odva.org/)

CAN Specifications – Controller Area Network, Version 2.0, 1991, Robert Bosch GmbH.
http://www.semiconductors.bosch.de/pdf/can2spec.pdf

10. Abbreviations

For the purposes of this chapter, the following abbreviations apply:

Abbreviation	Meaning
ASCII	American Standard Code for Information Interchange
CIP	The Common Industrial Protocol defined by Volume 1 of the CIP Networks Library
CID	Connection Identifier
EPR	Expected Packet Rate
ISO	International Standards Organization
MAC ID	Media Access Control Identifier (another name for Network Address)
OSI	Open Systems Interconnection (see ISO 7498)
UCMM	Unconnected Message Manager
CRC	Cyclic Redundancy Check
LED	Light Emitting Diode
MAC	Media Access Control sublayer
NAP	Network Access Port
NUT	Network Update Time
RG-6	Standard for coaxial cable
SMAX	MAC ID of the maximum scheduled node
UMAX	MAC ID of the maximum unscheduled node
FTP	File Transfer Protocol. An internet application that uses TCP reliable packet transfer to move files between different nodes.

Abbreviation	Meaning
RFC	Request For Comments (RFC) – This document series, which was launched in 1969, describes the Internet suite of protocols and related experiments. Not all (in fact, very few) RFCs describe the Internet Standards, but all Internet Standards are written up as RFCs. The RFC series is unusual in that the proposed protocols are forwarded by the Internet research and development community, acting on their own behalf, as opposed to the formally re-viewed and standardized protocols that are promoted by organizations such as CCITT and ANSI. [Source: RFC 1392]
TCP	Transmission Control Protocol (TCP) – An Internet Standard transport layer protocol defined in STD 7, RFC 793. It is connection-oriented and stream-oriented, as opposed to UDP. See also: connection-oriented, stream-oriented, User Datagram Protocol. [Source: RFC1392]
UDP	User Datagram Protocol (UDP) – An Internet Standard transport layer protocol defined in STD 6, RFC 768. It is a connectionless protocol which adds a level of reliability and multi-plexing to IP. See also: connectionless, Transmission Control Protocol. [Source: RFC1392]

11. Terminology

For the purposes of this document, the following definitions apply:

Term	Definition
allocate	In the DeviceNet context, this is the process of reserving resources of the pre-defined master/slave connection set in a DeviceNet node. It is associated with services of a similar name, of the DeviceNet object class (Class ID 0x03).
application	Typically refers to the application layer of the ISO-OSI model. The application layer is the part of the product that performs application-specific functions. Typically, application objects that provide the desired behavior are associated with the application.
attribute	A description of an externally visible characteristic or feature of an object. The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes also may affect the behavior of an object. Attributes are divided into class attributes and instance attributes.
behavior	Indication of how the object responds to particular events. Its description includes the relationship between attribute values and services.
bit	A unit of information consisting of a 1 or a 0. This is the smallest data unit that can be transmitted.
broadcast	A message that is sent to all nodes on a network. It also refers to the property of a network where all nodes listen to all messages transmitted for purposes of determining bus access/priority.
byte	A sequence of 8 bits that is treated as a single unit.
class	A set of objects, all of which represent a similar system component. A class is a generalization of the object, a template for defining variables and methods. All objects in a class are identical in form and behavior, but they may contain different attribute values.
object-specific service	A service defined by a particular object class to perform a required function that is not performed by any common services. A class-specific service is unique to the object class that defines it.
client	(1) An object that uses the services of another (server) object to perform a task. (2) An initiator of a message to which a server reacts.

Term	Definition
connection	A logical binding between two application objects. These application objects may be the same or different devices.
Connection Identifier	Identifier assigned to a transmission that is associated with a particular connection between producers and consumers that identifies a specific piece of application information.
connection path	Is made up of a byte stream that defines the application object to which a connection instance applies.
consumer	A node that is receiving (i.e., consumes) data from a producer.
consuming application	The application that consumes data.
CRC error	Error that occurs when the cyclic redundancy check (CRC) value does not match the value generated by the transmitter.
cyclic	Term used to describe events that repeat in a regular and repetitive manner.
datagram	A transmitted message.
device	A physical hardware connection to the link. A device may contain more than one node.
Device Profile	A collection of device-dependent information and functionality providing consistency between similar devices of the same device type.
drop or dropline	The cable that connects one or more nodes to a trunk cable, usually accomplished using a tap.
encapsulation	The technique used by layered protocols in which a layer adds header information to the protocol data unit (PDU) from the layer above for purposes of carrying one protocol within another.
Ethernet	A standard for LANs, initially developed by Xerox, and later refined by Digital, Intel and Xerox (DIX). In its original form, 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: IEEE 802.3, Local Area Network. [Source: RFC1392]
Expected Packet Rate	A misnomer, the Expected Packet Rate (EPR) is basic interval at which a connection transmits its data.

Term	Definition
fixed tag	A two-byte field in a ControlNet Lpacket that identifies unconnected or station management services the node is expected to perform. The first byte is the specific service code and the second byte contains the MAC ID of the destination node.
frame	See MAC Frame.
generic tag	A three-byte field in a ControlNet Lpacket that serves as the Connection Identifier (CID). It is associated with a specific piece of application information.
Guardband	It is the portion of ControlNet bandwidth that is allocated for the transmission of the moderator frame.
instance	The actual physical presentation of an object within a class. Identifies one of potentially many objects within the same object class.
Keeper	Object responsible for holding and distributing the Connection Originator schedule data for all Connection Originator devices on a ControlNet Network.
Link or Data Link	Refers to the Data Link layer of the ISO/OSI model.
Lpacket	On ControlNet, the Lpacket (or link packet) is a portion of the MAC Frame where application information that contains a size, control byte, tag, and link data is transmitted. There may be one or more Lpackets in a single MAC Frame.
MAC frame	A collection of MAC symbols transmitted on the network medium that contains the required message formatting/framing necessary to pass a message to another node. For example, a ControlNet MAC Frame consists of a preamble, start delimiter, source MAC ID, Lpackets, CRC, and end delimiter.
Message Router	The object within a node that distributes explicit message requests to the appropriate application objects.
multicast	A packet that is sent to multiple nodes on the network.
network	A series of nodes connected by some type of communication medium. The connection paths between any pair of nodes can include repeaters and bridges.
Network Access Port	On ControlNet, this is an alternate physical layer connection point on a permanent node that allows a temporary node to be connected to the link. The temporary node has its own network address, but simply shares the permanent node's physical layer connection to the network.

Term	Definition
network address	An integer identification value assigned to each node on a CIP Network.
network status indicators	Indicators (i.e., LEDs) on a node indicating the status of the Physical and Data Link Layers.
Network Update Time	Repetitive time interval on a ControlNet Network that is used to subdivide the network bandwidth. It determines the fastest rate that real-time data can be transferred on the network.
node	A connection to a link that requires a single MAC ID.
object	(1) An abstract representation of a particular component within a product. Objects can be composed of any or all of the following components: a) data (information which changes with time); b) configuration (parameters for behavior); c) methods (things that can be done using data and configuration). (2) A collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behavior.
object-specific service	A service defined by a particular object class to perform a required function that is not performed by one of the common services. An object-specific service is unique to the object class that defines it.
originator	The client responsible for establishing a connection path to the target.
point-to-point	A one-to-one data exchange relationship between two, and only two nodes.
port	A CIP port is the abstraction for a physical network connection to a CIP device. A CIP device has one port for each network connection. Within the EtherNet/IP specific context, a TCP or UDP port is a transport layer de-multiplexing value. Each application has a unique port number associated with it. [Source: RFC1392]
producer	A node that is responsible for transmitting data.
redundant media	A system using more than one medium to help prevent communication failures.
repeater	Two-port active Physical Layer device that reconstructs and retransmits all traffic on one segment to another segment.
scheduled	On ControlNet these are data transfers that occur in a deterministic and repeatable manner, on preconfigured NUT-based intervals.

Term	Definition
segment	This term has two uses within CIP. With respect to cable topology, a segment is a length of cable connected via taps with terminators at each end; a segment has no active components and does not include repeaters. With respect to explicit messaging, segments (logical segments, port segments, etc.) are used in explicit messages to describe various addressing elements of devices such as: class IDs, attribute IDs, ports, connection points, etc.
serial number	A unique 32-bit integer assigned by each manufacturer to every device. The number needs only be unique with respect to the manufacturer.
server	A device or object that provides services to another device (the client).
service	Operation or function that an object performs upon request from another object.
tap	Point of attachment on the trunk where one or more droplines are attached.
target	The end-node to which a connection is established.
terminator	A resistor placed at the physical extreme ends of trunk segments to prevent transmission reflections from occurring.
transceiver	The physical component within a node that provides transmission and reception of signals onto and off of the medium.
trunk or trunkline	The main bus or central part of a cable system, typically terminated at each end by a termination resistor.
Unconnected Message Manager	The function within a node that transmits and receives unconnected explicit messages.
unscheduled	On ControlNet, this refers to data transfers that use the unscheduled portion of the NUT.